IEC 61850-6

Edition 2.2 2024-11
CONSOLIDATED VERSION

INTERNATIONAL STANDARD

colour inside

Communication networks and systems for power utility automation – Part 6: Configuration description language for communication in power utility automation systems related to IEDs

IEC 61850-6:2009-12+AMD1:2018-06+AMD2:2024-11 CSV(en)

IEC Secretariat
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee, …). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC Products & Services Portal - products.iec.ch
Discover our powerful search engine and read freely all the publications previews, graphical symbols and the glossary. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 500 terminological entries in English and French, with equivalent terms in 25 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC 61850-6

Edition 2.2 2024-11
CONSOLIDATED VERSION

INTERNATIONAL STANDARD

colour inside

Communication networks and systems for power utility automation – Part 6: Configuration description language for communication in power utility automation systems related to IEDs

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 33.200

ISBN 978-2-8327-0071-6

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

COMMUNICATION NETWORKS AND SYSTEMS FOR POWER UTILITY AUTOMATION –

Part 6: Configuration description language for communication in power utility automation systems related to IEDs

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) IEC draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). [IEC/IEC and ISO] [takes/take] no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, IEC had not received notice of (a) patent(s), which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at https://patents.iec.ch or www.iso.org/patents. IEC shall not be held responsible for identifying any or all such patent rights.

This consolidated version of the official IEC Standard and its amendment has been prepared for user convenience.

IEC 61850-6 edition 2.2 contains the second edition (2009-12) [documents 57/1025/FDIS and 57/1041/RVD], its amendment 1 (2018-06) [documents 57/1918/FDIS and 57/1940/RVD] and its amendment 2 (2024-11) [documents 57/2711/FDIS and 57/2733/RVD].

International Standard IEC 61850-6 has been prepared by IEC technical committee 57: Power systems management and associated information exchange.

This second amendment constitutes a technical revision.

The main changes with respect to IEC 61850-6:2009+AMD1:2018 are as follows:

a) functional extensions concerning the engineering processto improve files exchange followup, SCL elements identification and control configuration handling, added;

b) provision of clarifications and corrections. Issues that require clarification are published in a database available at https://iec61850.tissue-db.com/. Arising incompatibilities are listed in 8.2.3.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all the parts in the IEC 61850 series, under the general title *Communication networks and systems for power utility automation*, can be found on the IEC website.

This IEC standard includes Code Components i.e. components that are intended to be directly processed by a computer. Such content is any text found between the markers <CODE BEGINS> and <CODE ENDS>, or otherwise is clearly labelled in this standard as a Code Component. In the current version of this document, such indication is made at the beginning of Annex A which identifies the list of XSD files and refers to the code component definition in Subclause 1.3.

The purchase of this IEC standard carries a copyright license for the purchaser to sell software containing Code Components from this standard directly to end users and to end users via distributors, subject to IEC software licensing conditions, which can be found at: http://www.iec.ch/CCv1.

This consolidated edition brings two distinct sets of changes:

1) Resolved Interop Issues (covered by the table below) which have already followed the technical issues (Tissues) process as described in IEC 61850-1 and have reached the green "status".

2) Resolved Editorial Tissues which may have led to interoperability issues.

The resolutions of these issues which lead to these changes are described in greater detail in the Tissue database hosted at https://iec61850.tissue-db.com/.

The only new features compared to the previous IEC 61850-6:2009+AMD1:2018 are the introduction of the UUID to identify elements and files, the modelling of controls binding from a client perspective, and the definition of translated labels for elements which may be represented in any user interface. Apart from this, this consolidated edition strictly respects the scope of the original edition.

Technical issues summary

N°, Subject, Clause and Paragraph are as they appear on the Tissue database hosted at https://iec61850.tissue-db.com/ where all technical issues have been stored from the origin of IEC 61850.

"Subject" defines very briefly the topic under focus.

The Tissues which have been considered are:

N°	Subject	Clause	Paragraph
1590	RCB: Offline changes increment ConfRev by 10000?	9.3.8	Table 23
1647	SDO@count definition inconsistent	9.5.3	Table 44
1648	DA@count definition needs restriction	9.5.4.1	Table 47
1669	Incorrect example of header	9.1	1
1672	Allow connection Server and ServerAt to the same SCL.Subnetwork	9.3.2	Below Table 50
1674	Harmonization with 62351-6	9.3.2	Services Element
1675	SCSM support capability - Harmonization with 62351-6	9.3.2	Services
1683	ICD file for IED functionality spanning for multiple VL and BAY	9.2.1	The name value is also a global identification of
1708	Presence of Sample Mode field not controllable through SmvOpts	9.3.11	Smv Options element
1729	Incorrect SCL example in (informative) Annex	D.2	2
1734	Improved schema validation	A.5	1
1740	Exceptions of enumeration types for IEC 61850-7-4	9.5.6	last in 9.5.6
1745	Definition of type and id in DataTypeTemplates not consistent	9.5.6	Table 49
1768	Server associate-request has no SCL parameters	9.3.2	Table 11
1771	SCL Services ReportControl max vs. Indexed	9.3.8	8
1774	Missing description of KDC	9.3.2	4
1786	Downgrade of SCD Exports not Mandatory	Annex G	Table G.2
1787	There is no clear mapping of all 7-2 ACSI type to SCL basic types	9.5.4.2	1
1808	Please clarify if ix first index is 0 or 1	9.3.6 Data object (DOI) definition	Table 19 and Table 20
1813	Typo "Valkind"	9.5.4.1	Table 46
1816	Add SICS statement for xsi:type usage in P elements	9.4.3 Annex G	7 Table G.1 and G.2
1818	Clarification of ExtRef attributes usage	9.3.13	Table 51
1823	Clarify iedType attribute usage in DataTypeTemplates	9.5.1	2
1831	ldInst reference should concretized	9.3.7	Table 22
1832	SICS I45 not clear enough	Annex G	Table G.1
1833	Service SettingGroups.ConfSG clarification	9.3.2	Table 11
1834	SICS I211 text not inline with Service section	Annex G	Table G.1
1839	Not clear definition of lnInst to LN0 type elements	9.3.5	5
1843	SCT handle different OriginalSclXxx and SCL version/revision/release	9.3.2 I.4.3.3	G.1
1854	SupSubscription	9.3.2	Table 11
1885	sAddr length	I.5.3.5	1
1886	Part 6 - Typo in Abbreviation	4	ICT

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn, or
- revised.

IMPORTANT – The "colour inside" logo on the cover page of this document indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

This part of IEC 61850 specifies a description language for the configuration of power utility IEDs. This language is called System Configuration description Language (SCL). It is used to describe IED configurations and communication systems according to IEC 61850-5 and IEC 61850-7-x. It allows the formal description of the relations between the utility automation system and the process (substation, switch yard). At the application level, the switch yard topology itself and the relation of the switch yard structure to the SAS functions (logical nodes) configured on the IEDs can be described.

While this part describes the language to describe the configuration of IEC 61850 systems, other parts of the standard describe how to configure the system and possible restrictions. Therefore implementations claiming conformance to this standard shall take into account constraints from the other normative references. Some references to the other parts have been included for the purpose of clarification but these references are not all inclusive.

NOTE The process description, which is in this standard restricted to switch yards and general process functions, will be enhanced by appropriate add-ons for wind mills, hydro plants and distributed energy resources (DER).

SCL allows the description of an IED configuration to be passed to a communication and application system engineering tool, and to pass back the whole system configuration description to the IED configuration tool in a compatible way. Its main purpose is to allow the interoperable exchange of communication system configuration data between an IED configuration tool and a system configuration tool from different manufacturers.

IEC 61850-8-x and IEC 61850-9-x, which concern the mapping of IEC 61850-7-x to specific communication stacks, may extend these definitions according to their need with additional parts, or simply by restrictions on the way the values of objects have to be used.

COMMUNICATION NETWORKS AND SYSTEMS FOR POWER UTILITY AUTOMATION –

Part 6: Configuration description language for communication in power utility automation systems related to IEDs

1 Scope

1.1 General

This part of IEC 61850 specifies a file format for describing communication-related IED (Intelligent Electronic Device) configurations and IED parameters, communication system configurations, switch yard (function) structures, and the relations between them. The main purpose of this format is to exchange IED capability descriptions, and SA system descriptions between IED engineering tools and the system engineering tool(s) of different manufacturers in a compatible way.

The defined language is called System Configuration description Language (SCL). The IED and communication system model in SCL is according to IEC 61850-5 and IEC 61850-7-x. SCSM specific extensions or usage rules may be required in the appropriate parts.

The configuration language is based on the Extensible Markup Language (XML) version 1.0 (see XML references in Clause 2).

This standard does not specify individual implementations or products using the language, nor does it constrain the implementation of entities and interfaces within a computer system. This part of the standard does not specify the download format of configuration data to an IED, although it could be used for part of the configuration data.

1.2 Published versions of the standard and related namespace names

The table below provides a reference between all published editions, amendments or corrigenda of this document and the full name of the namespace.

Edition	Publication date	Webstore	Namespace
Edition 1.0	2004-03	IEC 61850-6:2004	IEC 61850-6:2003
Edition 2.0	2009-12	IEC 61850-6:2009	IEC 61850-6:2007B
Amendment 1 of Edition 2.0	2018	IEC 61850-6:2009/AMD1:2018	IEC 61850-6:2007B4
Edition 2.1	2018	IEC 61850-6:2009+AMD1:2018 CSV	IEC 61850-6:2007B4
Amendment 2 of Edition 2.0	2024	IEC 61850-6:2009/AMD2:2023	IEC 61850-6:2007C5
Edition 2.2	2024	IEC 61850-6:2009+AMD2:2023 CSV	IEC 61850-6:2007C5

1.3 Identification of the namespace

The namespace associated with this document is an XML schema (XSD) for the System Configuration Language (SCL). The parameters which are identifying the namespace are provided in Table 53:

Table 53 – Attributes of the IEC 61850-6 XML namespace

Attribute	Content
Namespace nameplate	
Namespace Identifier (xmlns)	http://www.iec.ch/61850/2003/SCL
Version	2007
Revision	C
Release	5
XSD version header attribute	2007C5
Code Component Name	IEC_61850-6.SCL.2007C5.Full

1.4 Code Component distribution

Each Code Component is a ZIP package containing the electronic representation of the Code Component itself, with a file describing the content of the package (IECManifest.xml).

The life cycle of a code component is not restricted to the life cycle of the related publication. The publication life cycle goes through two stages, Version (corresponding to an edition) and Revision (corresponding to an amendment). A third publication stage (Release) allow publication of Code Component in case of urgent fixes of InterOp Tissues, thus without need to publish an amendment.

Consequently, new releases of the Code Component may be released, which supersedes the previous release, and will be distributed through the IEC TC57 web site at:

https://www.iec.ch/tc57/supportdocuments

The latest version/release of the code component will be found by selecting the file for the code component with the highest value for VersionStateInfo, e.g. IEC_61850-6.SCL.{VersionStateInfo}.full.zip.

The code component associated to this document is an XML schema file (XSD). It is available as a full version only. It is freely accessible on the IEC website for download at https://www.iec.ch/tc57/supportdocuments, but the usage remains under the licensing conditions.

In case of any differences between the downloadable code component and the IEC pdf published content, the downloadable code component is the valid one; it may be subject to updates. See included history files.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC TS 61850-2, *Communication networks and systems in substations – Part 2: Glossary*

IEC 61850-4, *Communication networks and systems for power utility automation – Part 4: System and project management*

IEC 61850-5, *Communication networks and systems for power utility automation – Part 5: Communication requirements for functions and device models*

IEC 61850-7-1:2011, *Communication networks and systems for power utility automation – Part 7-1: Basic communication structure – Principles and models*

IEC 61850-7-2, *Communication networks and systems for power utility automation – Part 7-2: Basic information and communication structure – Abstract communication service interface (ACSI)*

IEC 61850-7-3, *Communication networks and systems for power utility automation – Part 7-3: Basic communication structure – Common data classes*

IEC 61850-7-4, *Communication networks and systems for power utility automation – Part 7-4: Basic communication structure – Compatible logical node classes and data object classes*

IEC 61850-8-1, *Communication networks and systems for power utility automation – Part 8-1: Specific communication service mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3*

IEC 61850-9-2, *Communication networks and systems for power utility automation – Part 9-2: Specific communication service mapping (SCSM) – Sampled values over ISO/IEC 8802-3*

IEC IEEE 61850-9-3, *Communication networks and systems for power utility automation – Part 9-3: Precision time protocol profile for power utility automation*

IEC 62351-4, *Power systems management and associated information exchange – Data and communications security – Part 4: Profiles including MMS and derivatives*

IEC 62351-6, *Power systems management and associated information exchange – Data and communications security – Part 6: Security for IEC 61850*

IEC 62351-9, *Power systems management and associated information exchange – Data and communications security – Part 9: Cyber security key management for power system equipment*

IEC 81346-1, *Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations – Part 1: Basic rules*

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO/IEC 9834-8, *Information technology – Procedures for the operation of object identifier registration authorities – Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers*

RFC 1952, GZIP *file format specification version 4.3*, RFC, available at <http://www.ietf.org/rfc/rfc1952.txt>

RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, RFC, available at <http://www.ietf.org/rfc/rfc2045.txt>

Extensible Markup Language (XML) 1.0, W3C, available at <http://www.w3.org/TR/2000/REC-xml-20001006>

XML Schema Part 1: Structures, W3C, available at <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502>

XML Schema Part 2: Datatypes, W3C, available at <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>

3 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 61850-2 apply.

Additionally the following terms are used in the context of language name spaces. Only general meanings are given here. More details about the handling in the context of SCL can be found later in this standard.

3.1
extensible
a language is extensible if instances of the language can include terms from other vocabularies

Note 1 to entry: This is fulfilled in SCL if the other vocabularies come with their own XML name space.

3.2
language
an identifiable set of vocabulary terms that has defined constraints. For the purpose of this standard, the language is SCL.

Note 1 to entry: This is the case with SCL, although some constraints are not definable in the XML schema.

3.3
instance
a realization by usage of a language

Note 1 to entry: For example, an XML document in SCL describing an IED or a substation is an SCL instance.

3.4
sender
a tool that creates or produces an instance for processing by another application (receiver)

Note 1 to entry: SCL senders are typically IED and system configuration tools; e.g. the IED tool sends (produces) ICD files, the system tool sends SCD files.

3.5
receiver
a tool that consumes an instance which it obtained from a sender

Note 1 to entry: SCL receivers are IED tools and system configuration tools; e.g. the IED tool receives SCD files, the system tool ICD, IID, SSD and SED files.

3.6
processor
a component which receives SCL instances and produces new instances, i.e. is sender and receiver

Note 1 to entry: This is typically the system configuration tool.

3.7
project
a system part with engineering responsibility for all contained IEDs

Note 1 to entry: Mostly a system is a project. However, sometimes the IED engineering responsibility of different parts of a system belong to different parties or people. Each IED responsibility area is then a separate project. An IED can belong only to one project. It is 'owned' by this project.

3.8
backwards compatible
a language change is backwards compatible, if newer receivers can process all instances of the old language

Note 1 to entry: For SCL this means that tools built for newer language versions can understand instances from older versions. Especially system tools should understand old ICD and SSD files, while IED tools should understand old SCD files to be backward compatible.

3.9
forward compatible
a language change is forward compatible if older receivers can process all instances of the newer language

Note 1 to entry: For SCL this means that tools built according to older SCL versions can also process instances of newer SCL versions. Especially old system tools should handle new ICD and SSD files, while old IED tools should handle new SCD files to be forward compatible.

3.10
language version
the version of the XML schema defining the language

Note 1 to entry: A language instance is produced according to a language (schema) version, which is called its assigned version, although it may also be valid against other language versions.

4 Abbreviations

In general, the glossary and abbreviations defined in IEC 61850-2 apply. The following abbreviations are either exclusive to this standard, or particularly useful for understanding this standard and are repeated here for convenience.

BDA	Basic DATA Attribute (i.e. not structured)
CIM	Common Information Model for energy management applications
DAI	Instantiated Data Attribute
DO	DATA in IEC 61850-7-2, data object type or instance, depending on the context
DOI	Instantiated Data Object (DATA)
ICT	IED Configuration Tool
ID	Identifier
IED	Intelligent Electronic Device
ldInst	Instance identification of a Logical Device as part of its name
lnInst	Instance number of a Logical Node as part of its name
MSV	Multicast Sampled Value
MsvID	ID for MSV (Multicast Sampled Value)
RCB	Report Control Block
SCL	System Configuration description Language
SCT	System Configuration Tool
SDI	Instantiated Sub-DATA; middle name part of a structured DATA name
SDO	Sub-DATA within a DOType, referencing another DOType
SED	System Exchange Description
SST	System Specification Tool
UML	Unified Modelling Language according to http://www.omg.org/uml
URI	Universal Resource Identifier
UsvID	ID for USV (Unicast Sampled Value)
UUID	Universally Unique Identifier
XML	Extensible Markup Language

5 Intended engineering process with SCL

5.1 General

Engineering of a substation automation system may start either with the allocation of functionally pre-configured devices to switch yard parts, products or functions, or with the design of the process functionality, where functions are allocated to physical devices later, based on functional capabilities of devices and their configuration capabilities. Often a mixed approach is preferred: a typical process part such as a line bay is pre-engineered, and then the result is used within the process functionality as often as needed. For SCL, this means that it must be capable of describing:

a) a system specification in terms of the single line diagram, and allocation of logical nodes (LN) to parts and equipment of the single line to indicate the needed functionality;

b) pre-configured IEDs with a fixed number of logical nodes (LNs), but with no binding to a specific process – may only be related to a very general process function part;

c) pre-configured IEDs with a pre-configured semantic for a process part of a certain structure, for example a double busbar GIS line feeder, or for a part of an already configured process or automation system;

d) complete process configuration with all IEDs bound to individual process functions and primary equipment, enhanced by the access point connections and possible access paths in subnetworks for all possible clients;

e) as item d) above, but additionally with all predefined associations and client server connections between logical nodes on data level. This is needed if an IED is not capable of dynamically building associations or reporting connections (either on the client or on the server side).

Case e) is the complete case. Both cases d) and e) are the result after SAS engineering, while case a) is a functional specification input to SAS engineering, and b) and c) are possible results after IED pre-engineering either for a typical usage of the IED, or for a specific usage within a project.

5.2 Scope of SCL

The scope of SCL as defined in this standard is clearly focused on these purposes:

1) SAS functional specification (point 5.1 a) above),

2) IED capability description (points 5.1 b) and 5.1 c) above), and

3) SA system description (points 5.1 d) and 5.1 e) above).

These purposes shall provide standardized support to system design, communication engineering and to the description of readily engineered system communication for device engineering tools.

For practical purposes, the following is also supported:

1) exchange of system interfacing information between two projects handling two systems, which need to exchange data;

2) exchange of IED modifications on an IED instance engineered specifically for a project back from the IED tool to the system tool.

This is achieved by defining an object model describing the IEDs, their communication connections, and their allocation to the process functions and equipment, as well as a standardized way to describe how this model shall be represented in a file to be exchanged between engineering tools. The resulting object model could also be the base for other engineering tasks, possibly with some additions. Therefore, and because of the additional needs of SCSMs, this standard considers the language as defined here as the core model, and defines how extensions of this core model for SCSMs as well as other (engineering) purposes can be carried out in a standardized and compatible way. IEC 61850-4 gives an overview about the complete system life cycle.

5.3 Use of SCL in the engineering process

Figure 1 explains the usage of SCL data exchange in the above-mentioned engineering process. This is just a part of the overall engineering process and the complete system life cycle as described in IEC 61850-4. The text boxes in Figure 1 above the dashed line indicate where SCL files are used. The text box *IED capabilities* corresponds to a result of steps 5.1 b) and 5.1 c) above, the text box *System specification* corresponds to step 5.1 a) above, the text box *Associations*… refers to steps 5.1 d) or 5.1 e) above.

To make the engineering tasks and responsibilities clear, tool roles are introduced for an IED configurator and a system configurator. Clause 10 provides more details about the engineering rights of the tool roles. A tool implementation can have both roles. In this case the transfer of partly engineered data within the tool is private, but to any other (mostly to an IED tool) it has to be seen from the role the tool has played when modifying the project data, i.e if the modification was done in the scope of an IED tool, or in the scope of a system tool.

The **IED Configurator (ICT)** is a manufacturer-specific, may be even IED-specific, tool that shall be able to import or export the files defined by this part of IEC 61850. The tool then provides IED-specific settings and generates IED-specific configuration files, or it loads the IED configuration into the IED.

An IED claiming conformance to the IEC 61850 series shall fulfil the following:

- it is accompanied either by an (ICD) SCL file describing its capabilities, or by an (IID) SCL file describing its project specific configuration and capabilities, or by a tool, which can generate one or both of these file types from or for the IED (not shown in Figure 1);
- it can directly use a system SCL (SCD) file to set its communication configuration, as far as setting is possible in this IED (i.e. as a minimum, its needed communication addresses), or it is accompanied by a tool which can import a system SCL file to set these parameters to the IED.

The **System Configurator (SCT)** is an IED independent system level tool that shall be able to import or export configuration files defined by this part of IEC 61850. It shall be able to import configuration files from several IEDs of one or several manufacturers, as needed for system level engineering, and used by the configuration engineer to add system information shared by different IEDs. Then the system configurator shall generate a project-related configuration file as defined by this part of IEC 61850, which is fed back to the IED Configurator for system-related IED configuration. The System Configurator should also be able to read a System specification file for example as a base for starting system engineering, or to compare it with an engineered system for the same process part.

The **System Specification Tool (SST)** is an implementation independent system level tool that shall be able to create a full system topology without the need to integrate real devices. It produces a System specification file to be used by the System Configurator as a base for a new system or as a template.

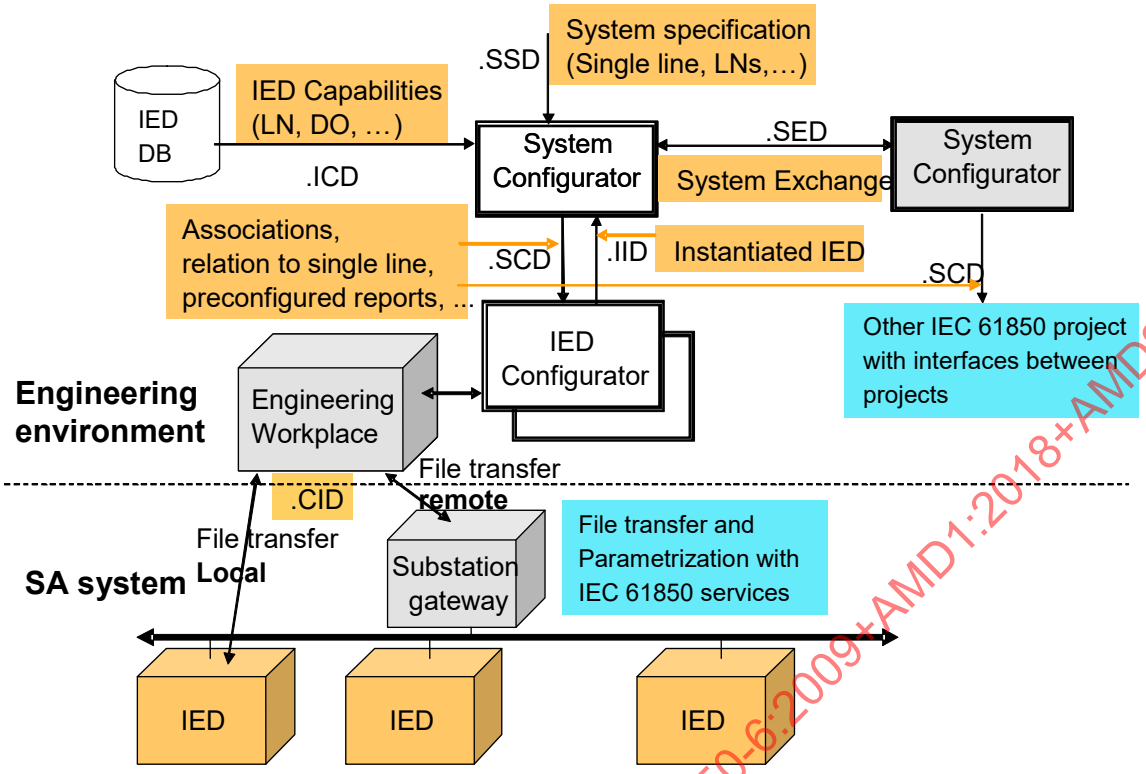


Figure 1 – Reference model for information flow in the configuration process

The part of Figure 1 below the dashed line indicates the ways in which IED configuration data produced by means of the IED configurator can be brought into the IED. This can be effected by:

- local communication from an engineering workstation connected locally to the IED. This data transfer is beyond the scope of this standard.
- remote file transfer, for example by the file transfer method of IEC 61850-7-2. The file format is not defined within this standard, but SCL format is a possible choice at least of a part of the configuration data.
- access services to parameter and configuration data defined according to IEC 61850-7-2. In this case, the standardized methods according to IEC 61850-7-x shall be used.

NOTE It is not in the scope of this standard to define any details of concrete software tools, which support an engineer in doing the intended engineering process with SCL as described above. Both the system configurator as well as the IED configurator introduced above are conceptual tools, respectively tool roles to illustrate the use of different SCL file variants in the engineering process. Each manufacturer is completely free to find the best way to support engineers by a specific software tool. In addition, complete freedom of choice is given in the way in which software tools for the above described engineering process with SCL will store manufacturer-specific internal parameters for IEDs and SA system aspects not covered by the scope of IEC 61850 (e.g. the relation of logical data to pins on a physical board), and how they relate them to the IEC 61850 data model.

Figure 1 gives a static view of file type exchange between tool roles. The sequence of data exchanges between engineering tools during the engineering process can then be as follows (see Figure 2):

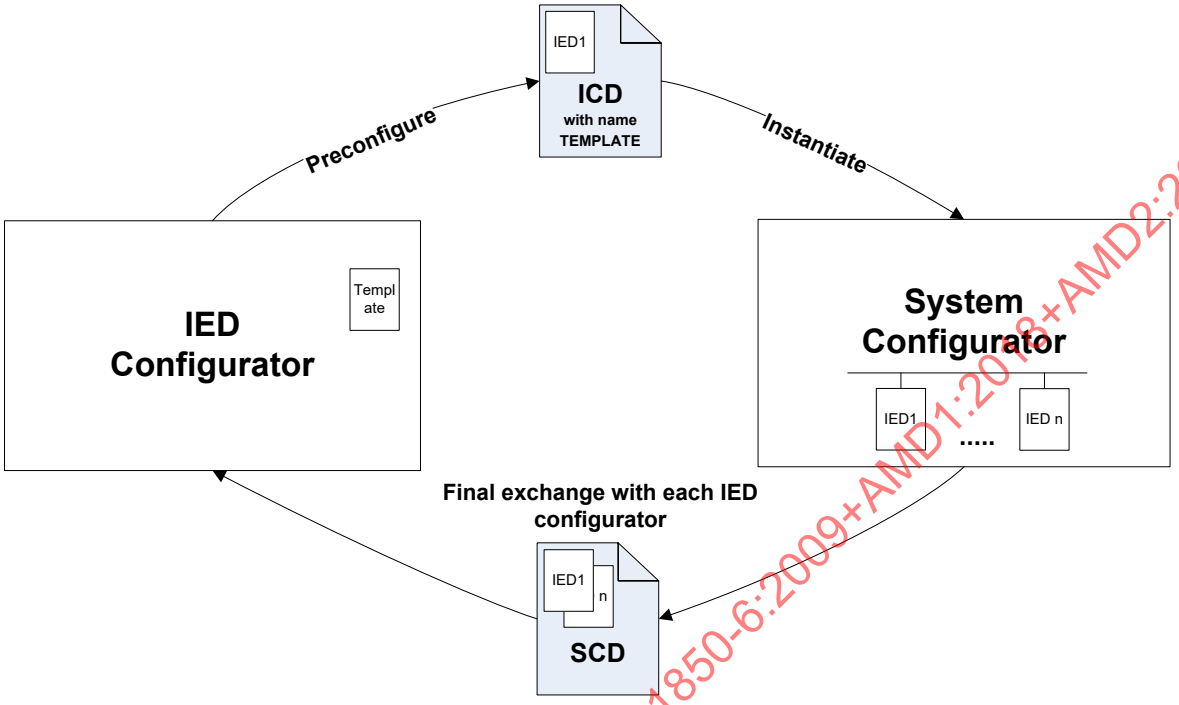


Figure 2 – IED type description to System Configurator

At start of system engineering the IED capability (ICD) files are used by the system configurator as IED template (type) description to instantiate project specific IEDs as needed. As Figure 2 shows, this ICD file can be generated in advance as typical configuration of an IED by means of the IED configurator tool, as typically done for very flexibly configurable IEDs.

Alternatively the system configurator can also import the description of an IED specifically preconfigured with name and addresses for a concrete function in the process by importing an Instantiated IED Description (IID) file, as indicated in Figure 3.

The SCD file generated by the system configurator tool is then imported by the IED configurator for the final IED instance configuration, as shown in Figure 2 and Figure 3. Any add-ons or adapted values could then be exported by means of the IID file, and thus brought back to the system configurator respective the next revision of the SCD file – see next subclause on IED modifications.

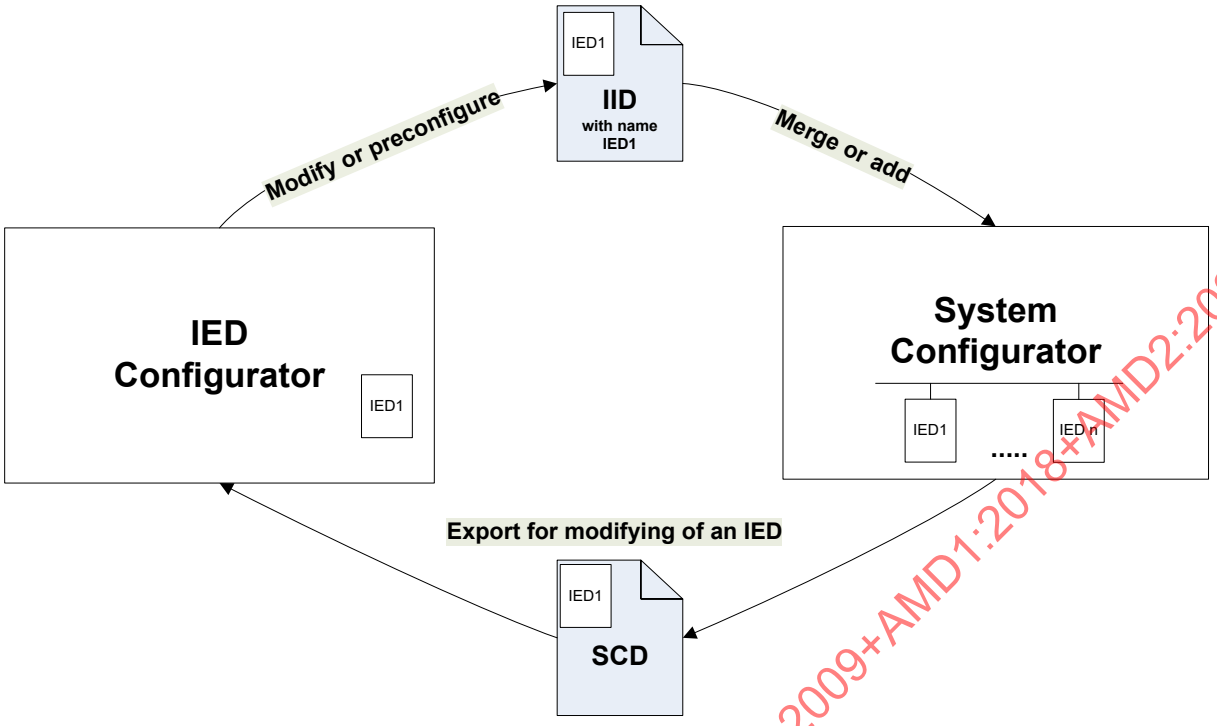


Figure 3 – IED instance description to System Configurator

5.4 IED modifications

During the engineering process it may happen that the IED-related data has to be changed. This can in principle be done by removing the IED from the system, and reinstantiating a modified IED description file in the system. However, in this case also all existing references from or to the IED are lost and have to be re-established. On the other hand, tool responsibilities shall be clarified as follows:

The IED configurator is responsible for the IEDs data model, and all its configuration values. It is not allowed to change any data flow- and communication-related definitions coming from system engineering. To assure this, it shall not modify a system description (SCD) file.

The System configurator is responsible for the communication addressing and the data flow between the IEDs, within the scope of the IED capabilities. It might set configuration and parameter values as needed from the system point of view. It is not allowed to change the IEDs data model.

Therefore another SCL file is defined, allowing to update the IED data within a system. It is called IID file (Instantiated IED Description), and describes the project specific configuration of an IED. In the case that this IED does not exist in the SCD file, it can be imported completely and instantiated as a project specific IED, without any references to other IEDs (see also Figure 3). In the case that it exists already, the data model part inclusive any values can replace the appropriate parts existing in the system configurator. All data flow-related definitions and references to other IEDs, which exist in the system configurator, shall still be valid. The IED configurator is not allowed to make changes which modify the already configured data flow and communication addresses. Especially, no names shall be changed, and no referenced parts of the data model shall be deleted. The general process is shown in Figure 4.

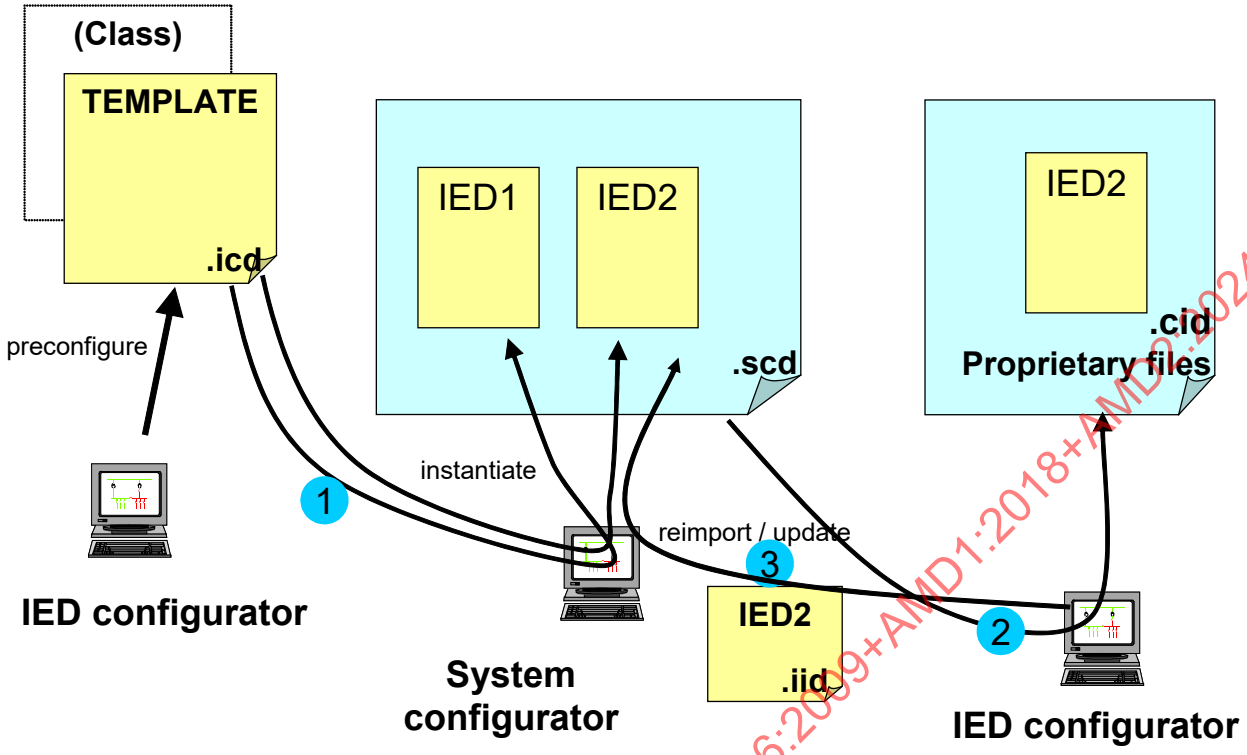


Figure 4 – Modification process

At the start of system engineering the IED capability (ICD) files are used by the system configurator to instantiate project specific IEDs as needed (1). The resulting SCD file is then imported by the IED configurator for the final IED configuration (2). Any add-ons or adapted values could then be exported by means of the IID file, and thus brought back to the system configurator respective the next revision of the SCD file (3).

To allow detection of a modified file, the IED owner within the IID file should be set to the header identification of the SCD file, which was the input to IED engineering before.

5.5 Data exchange between projects

As far as the engineering responsibility is concerned, a complete secondary system can be split into different parts. Examples include but are not limited to separate engineering of high-voltage level and medium-voltage level, of a transformer-related part, of separate bays, or even of different substations exchanging data e.g. for line protection or interlocking. For the purposes of this standard, such a system part with responsibility for all its contained IEDs is called a **project**. To allow the engineering of online communication data flow between such projects, some interfacing data has to be exchanged between the projects, and the engineered interfaces have to be reimported to the concerned projects.

NOTE 1 From a statical point of view a *project* can be considered as a responsibility area. As the exchange of information between responsibility areas is carried out in processes, the term *project* is used here.

To facilitate this engineering data exchange, the following rules are set up.

a) An IED always belongs to a project. Only people and tools belonging to the project are allowed to configure the IED, especially handling of data transfer between the project data base of the system tool and the IED tool. The owner project has full engineering rights on the IED.

b) A project can transfer to another project the right to add definitions for data flow from some of its IEDs (IED check out). This has to be accompanied by a description of those parts of the IED which are allowed to be used and enhanced by the other project. This transfer of 'data flow' engineering rights blocks any modification of the exported IED in the owner project. Before this IED can be modified within the owner project, the 'data flow' engineering right has to be transferred back to it (IED check in), normally with some added data flow definitions.

c) To not lose already engineered references on such exported IED parts, parts of referenced IEDs have also to be exported as fix IEDs. These are not allowed to be changed by the importing project, and must be reexported unchanged when the ready engineered IED is transferred back to the owner project.

d) Needed parts of the Substation section and communication section shall be exported. From the Substation section only full bays shall be exported, although they shall contain only LN links to exported IEDs. The importing project is only allowed to add logical node links to the substation, or add a part of its own substation section. Furthermore, it is allowed to add addresses in the existing subnetworks and own subnetworks to the communication section. It is up to the project engineer to ensure that objects entered by him also have unique names in the exporting project, and that the entered addresses are unique within the full Subnetwork.

The transfer of rights occurs by means of an SCL file, called a SED (System Exchange Description) file. This file also contains the transferred engineering rights (fix, or dataflow) and the IED engineering capabilities (e.g. number of data sets and control blocks allowed to be used by the receiving project).

The parts of an IED exported with dataflow rights shall be frozen for engineering at the IED owner project (after IED check-out), until the using project transfers it back by means of another SED file (IED check-in). The file transferred back shall have the same SCL Header identification as the originally exported file but with an increased revision index.

The Header identification of an SCD file is taken as project identification, and therefore it is always different to that of a SED, which identifies an engineering data transfer between exactly two projects. The SCD Header identification is used within a SED file to identify the owner project of an IED.

The transfer and handling of the *dataflow* and *fix* rights is defined in the state diagrams of Figure 5 for the project owning the IEDs as well as for the receiving project. Observe that the *full* engineering right always stays in the owner project.

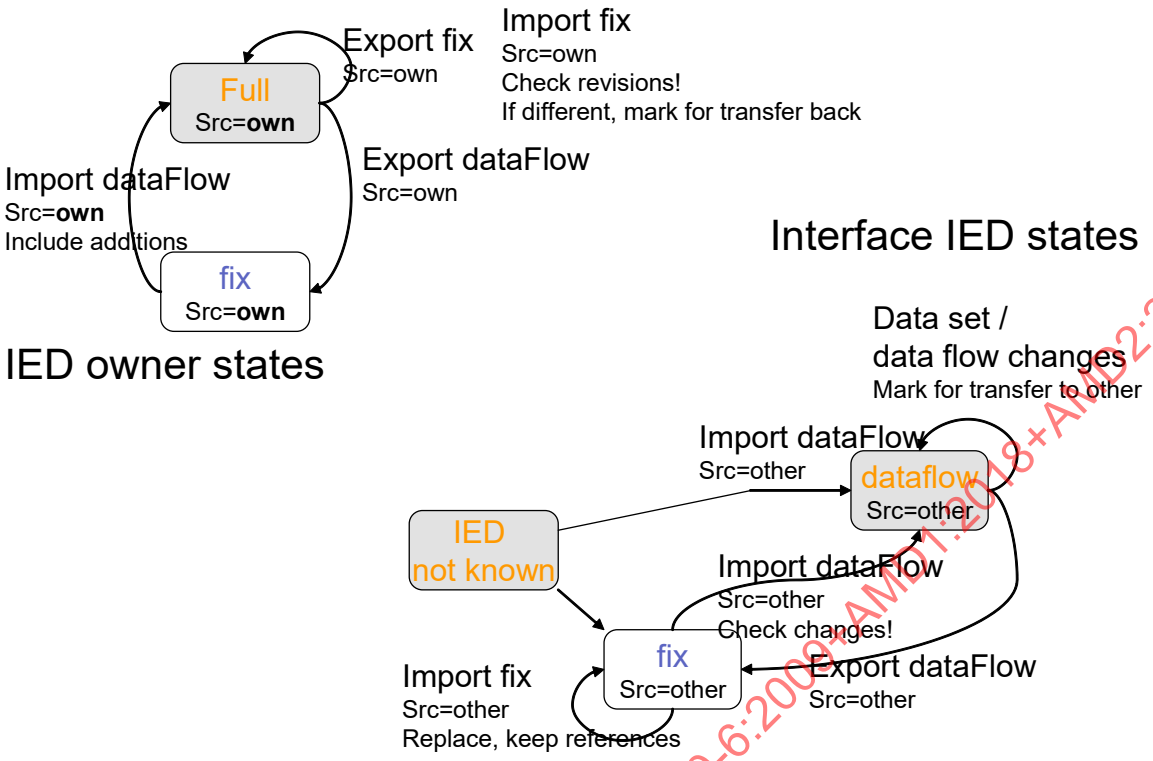


Figure 5 – Engineering right handling in projects

The state diagram on the left shows the internal states in the IED owner project, when exporting and importing owned IEDs via SED file. The right / lower state diagram shows the IED states of imported / exported IEDs at the receiving project, which is not the owner of the IEDs. The src property defines whether the IED owner is the own project or some other project.

Example: the owner project exports an IED with engineering right fix. In this case itself contains full engineering rights – however also to inform the other project in case something relevant for it is changed. The other project then imports this IED. As the IED right is fix, it is not allowed to change anything at the IED, however it will use it as destination IED for some of its own IEDs which need to send data to it. When ready, the imported IED as well as any IEDs from the other project sending data to it are exported as SED file and imported in the owner project, which now can complete the IED engineering to receive the data from the other project.

The detailed engineering rights in terms of SCL elements are described in Clause 10, after all elements have been introduced. It is recommended, that the temporary fix state of an IED with exported dataflow engineering rights is also shown in the SCD file, if a SCD is produced from the project at that time.

If several (system) tools are used within the same project, it is the responsibility of the engineer(s) to use them in such a way, that the system description generatable as an SCD file stays consistent. It is a project internal issue if in this case also the rights transfer mechanism is used within the project.

NOTE 2 The engineering rights transfer can be considered as a checking out of the IED to a specific project, with later checking in again, after the data flow modifications have been done in the other project. Only one (other) project at a time is allowed to engineer the IED.

6 The SCL object model

6.1 General

The SCL language in its full scope allows to describe a model comprising:

- the primary (power or process) system respective function structure: which primary apparatus functions are used, and how the apparatus are connected. This results in a designation of all covered primary equipment like switchgear as utility (automation) functions, structured according to IEC 81346-1;
- the communication system: how IEDs are connected to subnetworks and networks, and at which of their communication access points (communication ports) with which addresses;
- the application level communication: how data is grouped into data sets for sending, how IEDs trigger the sending and which service they choose, which input data from other IEDs is needed;
- each IED: the logical devices configured on the IED, the logical node instances with class and type belonging to each logical device, the reports and their data contents, the (pre-configured) associations available; and which data shall be logged; further IED engineering capabilities;
- instantiable logical node (LN) type definitions. The logical nodes as defined in IEC 61850-7-x and other domain standards have mandatory and optional DATA (here abbreviated DO, DATA objects) as well as optional services, and are therefore not instantiable. Further it is allowed to add user defined DATA. In this standard therefore instantiable LNTypes and DOTypes are defined as templates, which contain the really implemented DOs and services;
- the relations between instantiated logical nodes and their hosting IEDs on one side and the process 7 switch yard (function) parts on the other side.

SCL allows the specification of user defined DOs as an extension of standard LN classes as well as completely user-defined LNs according to the rules of IEC 61850-7-1. This means that the appropriate name space attributes shall be defined in the logical node types, and their value shall appear in the SCL file.

An SCL file describes an instance of the model in a serialized form and standardized syntax. However its semantic can only be fully understood by reference to the model itself, i.e. it is independent from the syntax. Clause 6 therefore gives an overview of the model by using UML notation. The following clauses then define how an instance of the model is formally described in SCL.

The UML object model is contained in Figure 6. Note that it is not complete in the modelling sense, i.e. it does not show any superclasses from which the used classes may be derived, or any attributes. It restricts itself to those concrete object types that are used within a SCL instance file, in the case of the substation-related part, mainly for the purpose of functional designation. Furthermore it does not contain the levels below DATA (DOs), which are structurally defined in IEC 61850-7-2 and whose SCL description is defined in the DataTypeTemplates clause.

The object model has three basic parts:

1) Substation / Line / Process: this part describes the primary process related functions and devices like switch yard, respectively any primary process in the functional view according to IEC 81346-1, electrical connections on single line level (topology), and the designation of equipment and functions;
2) Product: this stands for all SA product-related objects such as IEDs and logical node implementations;
3) Communication: this contains communication-related object types such as subnetworks and communication access points, and describes the communication connections between IEDs as a base for communication paths between logical nodes as clients and servers.

Additionally, the data type template section allows, in a type-oriented (i.e. reusable) way, the specification of which data and attributes really exist in an IED. A logical node type as specified there is an instantiable template of the data of a logical node.

More model details contained in SCL, for example the structure within the logical nodes, are described in IEC 61850-7-x.

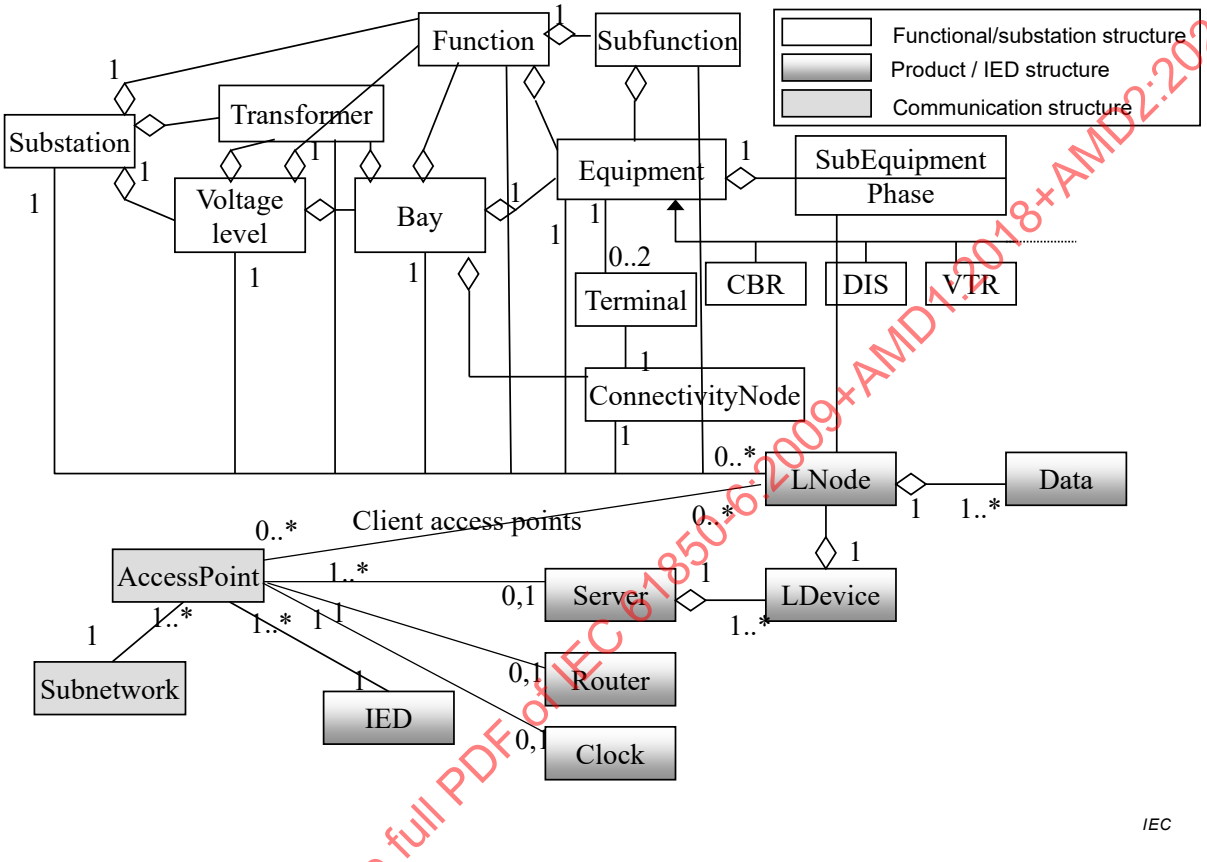


Figure 6 – SCL substation object model

The process / substation part and the product part in itself form hierarchies, which are used for naming and can be mapped to the functional and product structures according to IEC 81346 (all parts). The communication model part just contains the communication connection relations of IEDs to subnetworks, between subnetworks by means of routers at an IED, and the placement of master clocks at the subnetworks for time synchronisation. The modelling of gateways is not especially considered. A gateway which is an IEC 61850 server has to be modelled like any other IEC 61850 compliant IED. The Proxy DO in the LPHD logical node makes it possible to specify whether a hosted LD is an image of another IED, or belongs to the hosting IED. A gateway being an IEC 61850 compliant client should host an ITCI logical node.

As Figure 6 shows, the logical node (abbreviated as LN or LNode) is the transition object which is used to connect the different structures. This means that the LN instance as a product also has a functional aspect within the switch yard functionality and a communication aspect as a client or as a server within the substation automation system.

The process / substation functional objects as well as the product-related objects are hierarchically structured. Each higher level object consists of lower level objects. This hierarchy is reflected in the designation structure of the objects according to IEC 81346-1. The function structure of IEC 81346-1 shall be used, and the designation coding of IEC 81346-2 should be used in the process / substation objects, while the IEC 81346-1 product structure should be used for IED designation structure and the IEC 81346-2 codes for the name values.

In SCL, it is foreseen that within each structure for nearly all objects, two kinds of designation are possible.

- A name is used as (a hierarchical part of) a technical key to designate the object. Each object within a hierarchy has an attribute *name*, which contains its identification within this level of the hierarchy. Technical keys are used in technical documentation for building and maintaining the system, or for automatic processing of engineering-related information. This designation is also used in SCL to describe links between different model objects. In this case, as far as possible, the attribute containing the link gets a name of the form *<targettype>Name,* for example *daName* for a link to a DATA attribute. This *name* relates to and is mostly identical to what is called *name* in IEC 61850-7-2.
- A description part is used as (a hierarchical part of) an operator- or user-related object identification. An object within a hierarchy has an attribute *desc*, which contains its textual description part within the hierarchy. Textual identifications are for example used in operator interfaces and operator manuals.

NOTE The desc SCL attribute is used at engineering time, and identifies a (functional) object at its hierarchy level to a human being. The IEC 61850 d DATA attribute is used for describing data, and could also be read online. The contents of desc attributes could be used to generate a project specific (SCD) d text from a template (ICD) d text. This is however not standardized.

A reference within SCL is, as defined in IEC 61850-7-2, a unique identification of an object, containing as a path the concatenation of all names in the hierarchy levels above, up to the level of the object. For the connection of power system equipment within a single line diagram, this path is used explicitly, while for other references it is used implicitly by stating only missing name parts. For forming names according to IEC 61850-7-2, the term *instance* with the abbreviation *inst* is also used. It is a part of a IEC 61850-7-2 name, making the full name (path name) unique within this level (see examples in 8.5).

In addition to the full path used to identify any object in the SCL by its name, the SCL introduces the UUID (Universal Unique Identifier) which can be used to identify objects independently of their name which can evolve all along the lifecycle of a system. The reference to a UUID can be inside the SCL file itself or outside to be used during the external process not only dealing with SCL files (e.g. as per requirement from IEC 61850-6-2 for human machine interface engineering).

The following subclauses describe the different parts of the model, their meaning and respective usage. Object attributes are mentioned here only if necessary for the understanding of the model. Further object attributes are described later in the SCL definition. Further model details belonging to IEC 61850-7-x and especially explained in IEC 61850-7-1 and IEC 61850-7-2 are purposely not shown here. The name model of the switch yard functionality is however only found in this part of IEC 61850, and therefore shown as far as it is used within this part of IEC 61850.

Figure 7 shows an instance of this model: a simple example of a SA system used for a switch yard. The naming is performed according to the IEC 81346 series. The switch yard has a 110 kV voltage level E1. It is a double bus bar system with two line bays =E1Q1 and =E1Q3, and a bus coupler =E1Q2. The IEDs are already assigned to switch yard functionality (for example the bay controller -E1Q1SB1 as a product is assigned to bay =E1Q1, and its LN CSWI1 controls the circuit breaker =E1Q1QA1 via the LN XCBR1 on the IED -E1Q1QA1B1). Observe that in IEC 81346-1 terms, in this instance the bay is a transition object, i.e. it has a function (= sign, at switch yard level), and it is considered to be a part of the switch yard as a product. This transition can be seen in an SCL description in the name structure of the IED name. Only the transition at the logical node is modelled explicitly. Figure 7 shows with the – (Minus) sign the product-related designation. The functional name is not repeated. The station level communication subnetwork is named W1. There are three additional subnetworks at process level (W2, W3, W4). Access points are seen in the picture, but their designations are not shown. Logical devices and servers are also not shown in the picture. This means especially that dynamic connections such as associations are not shown.

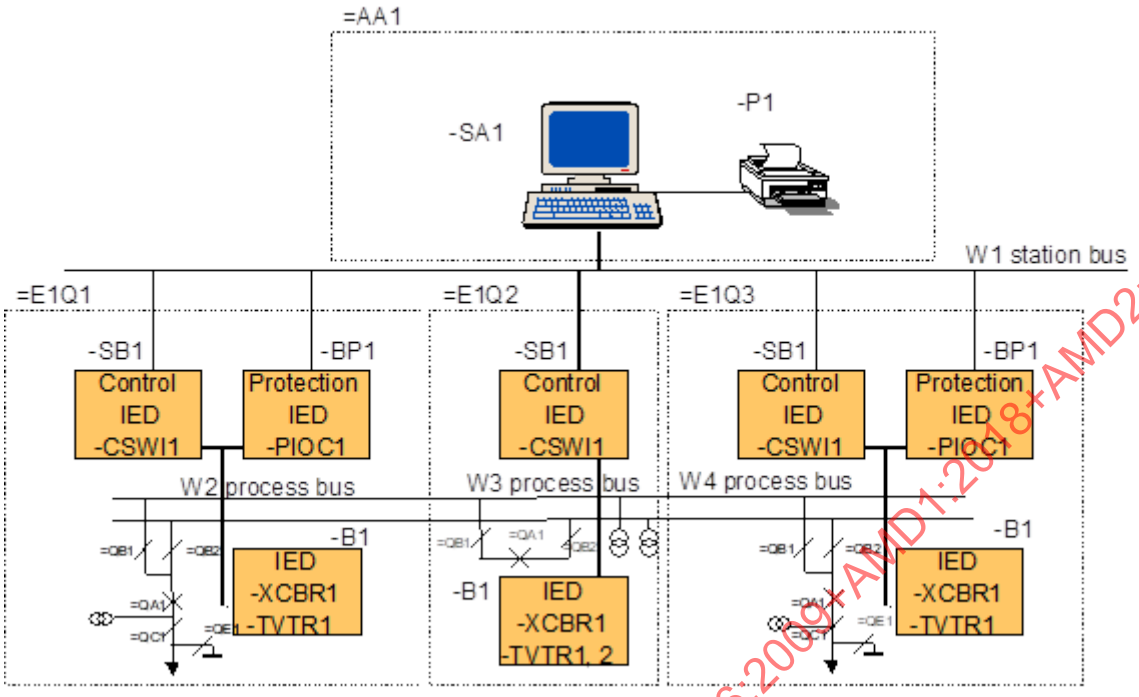


Figure 7 – SA System Configuration example

6.2 The process model

The process model (upper part of Figure 6 for substations) is an object hierarchy based on the functional structure of the primary process. A special primary process is the Substation, which has its own SCL element. Another is the Line, which connects electrically bays of substations. The generic *Process* element shall be used to model any additionally needed structuring levels of a power grid above the substation and for all other primary processes. Although each object is self-contained, its reference designation is derived from its place in the hierarchy. Because LNs perform functions within the complete context of the Process / Substation respective Line hierarchy, they can be attached as functional objects at each function level. Typically, a switch controller LN is attached to a switching device, while a measuring LN is attached to the bay, which delivers the measurands, and transformer-related LNs are attached to the appropriate transformer.

NOTE 1 In the CIM model measurands are allocated to primary device terminals. This is a topological allocation, while the allocation in SCL in first line serves functional naming. However, if the single line topology is modelled completely by means of the transformers (VTR, CTR) and their data acquisition nodes (TVTR, TCTR) also some primary device terminal in the topology can be found to which the measurands belong according to the CIM model.

The purpose of the process model is

- to relate a logical node and its function to a function of or at the primary process (process part or substation part or line part or equipment or subequipment);
- to derive a functional designation for the logical node from the process structure.

The following objects of the functional structure (in hierarchical order) are used in the SCL model, for power networks analogue to the CIM model for energy management systems. More background information on these terms can be found in IEC 61850-2:

Process — the whole or part of the primary process handled by the automation system, which is no substation. Can be parts of the power grid containing several substations, or complete other primary processes like power plants.

Line	a line connecting several substations.
Substation	the object identifying a whole substation.
VoltageLevel	an identifiable, electrically connected substation part having an identical voltage level.
Bay	an identifiable part or subfunction of the switch yard (substation) within one voltage level.
Equipment	an apparatus within the switch yard, for example circuit breaker, disconnector, voltage transformer, power transformer winding etc. The single line diagram of a switch yard shows the electrical connections between these primary devices. Connectivity node objects model these connections. Therefore, each primary device can contain at its terminals references to the connectivity nodes to which it is connected. At single line level, one or two terminals (connections) per equipment are normally sufficient.
SubEquipment	a part of an Equipment, which might especially be one phase of a three-phase equipment.
ConnectivityNode	the (electrical) connectivity node object connecting different primary devices. Typical connectivity node examples are: connecting nodes within a bay, bus bars connecting several bays in the same voltage level, lines connecting bays in different substations. See also Equipment above.
Terminal	an electrical connection point of a primary apparatus at single line level. A terminal can be connected to a ConnectivityNode. Within SCL terminals can be explicitly named, or exist implicitly.
Function	allows additional functions at substation, voltage level or bay level, either independent from the basic switch yard functionality like fire fighting or building supervision, or as part of the switch yard like main 1 protection and main 2 protection.
SubFunction	a hierarchical subpart of a Function or SubFunction, e.g. earth fault protection as subpart of the main 1 function.
EqFunction	allows additional functions at or below Equipment level, e.g. redundant functions on the same equipment.
EqSubFunction	a hierarchical subpart of an EqFunction or EqSubFunction.

The *PowerTransformer* is special equipment, which can hierarchically be located below Substation, VoltageLevel or Bay. It contains Transformer windings as equipment, which might again have a relation to a tap changer.

NOTE 2 Observe that the hierarchical structure is used for functional designations. If substructures of bays are needed, this can be introduced by appropriate structured bay names. If, for example, a bay B1 is structured into sub-bays SB1 and SB2, this would in the SCL model lead to two bays named B1.SB1 and B1.SB2. If logical nodes are also attached to the B1 structure level, then B1 can be introduced as a third bay.

NOTE 3 In the CIM model the bay level is optional, while in SCL it is mandatory. However, if the bay level structuring is not needed, a whole voltage level can be considered to be one bay. The only restriction here is that the SCL syntax demands at least one character as name on each level, so that in this case the voltage level name needs at least 2 characters, from which within the SCL substation structure the first character is taken as the voltage level name, and the last character is taken as the name for the one bay element.

6.3 The product (IED) model

Products consisting of hardware or software implement the functions of the switch yard. The scope of SCL from the product side only covers the hardware devices (called IEDs) that form the substation automation system, and therefore restrict the model to them. Primary devices as products are outside the scope of SCL, only their functional side is modelled by the substation structure for functional naming purposes.

IED	a utility automation device performing automation functions by means of logical nodes (LNs). It normally communicates via a communication system with other IEDs in the automation system.

Server	a communication entity within an IED according to IEC 61850-7-x. It allows access via the communication system and its only access point to the data of the logical devices and logical nodes contained in the server.
LDevice	a logical device (LD) according to IEC 61850-7-2 that is contained in a server of an IED.
LNode	a logical node (LN) instance according to IEC 61850-5 and IEC 61850-7-2, contained in a logical device of an IED. The LN contains Data (DO), which other logical nodes request, and it may need DOs contained in other LNs to perform its function. The *offered DOs* (server capability) are described in SCL. The *needed DOs* (LN client side) are determined by the function (LN) implementation and therefore configured by the IED configurator respective by the engineer, which plans the system. SCL also allows their description, so that a data flow on data level between LNs can be modelled.
DO	the DATA contained in the LNs according to IEC 61850-7-x.

NOTE Figure 6 shows with its LNode class the LN object, whose instances can be referenced or represented in SCL in two ways. The *LNode* element resides in the Substation structure, while the *LN* element resides in the IED structure.

6.4 The communication system model

The communication model is, in contrast to the others, not a hierarchical model. It models the logically possible connections between IEDs at and across subnetworks by means of access points. A subnetwork is seen at this description level only as a connecting node between access points, not as a physical structure. A logical device or a client of an IED is connected to a subnetwork by means of an access point, which may be a physical port or a logical address (server) of the IED. Client LNs use the address attribute of the access point to build up associations to servers on other IEDs respective to the LNs contained on the logical devices of these IEDs. Subnetworks may be connected by routers, however GSE messages as well as SV / SAV messages can not cross routers and can only reach IEDs within the same subnetwork. For accurate time synchronisation further each subnetwork should have an own (master) clock connected.

Although subnetworks only model logically possible connections, a correlation to the physical structure can be built up by appropriate naming of subnetworks and access points, and by the relation of access points to (one or more) physical connection points. The access points are the matching elements (transition objects) of both this communication model and the physical implementation of the communication system. The description and maintenance of the physical structure is beyond the scope of SCL, although some features allow to model it at least partly – see also 9.4.6.

This standard introduces as additional IED functions:

- a *Router* function on an IED. An IED with a router function can be connected with two different access points to two different subnetworks and allow TCP-based messages to reach IEDs within the other subnetwork;
- a *Clock* function to indicate where a subnetwork master clock is located.

Furthermore, the IED type SWITCH is reserved to model arbitrary switch based Ethernet networks, e.g. for IP address checking or modeling of the physical network. IEDs of type SWITCH typically consist only out of an access point to their IP subnetwork. This type SWITCH is stated by means of the IED type attribute. IED type designations of the really used switches can be used instead, if known.

Subnetwork — a connecting node for direct (link layer) communication between access points. It might contain telegram filtering on the bridge level, but no routing on the network level. All access points connected to a subnetwork can communicate with all others on the same subnetwork with the same protocol. SCSMs may define restrictions to this, for example if the stack implements a master-slave bus. The subnetwork as used here is a logical concept. Several logical subnetworks with different higher layer protocols could for example be used on the same physical bus to allow mixing of higher-level protocols on the same physical (lower) layer(s).

Access point — a communication access point of the logical device(s) of an IED to a subnetwork.. An access point may serve several logical devices, and the logical nodes contained in a logical device may, as clients, use several access points to connect to different subnetworks. Typically, a switch controller LN may receive data as a client from a process bus, and provide data as a server to the inter-bay bus (IEC 61850-8-1). In the terminology of IEC 61850-7-x, an access point may be used by a server, by a client, or by both. Furthermore, the same (logical) access point might support different physical access ports, for example an Ethernet connection and a serial PPP-based connection to the same higher level (TCP/IP) access point and to the same server.

Router — Normally, clients connected to a subnetwork only have access to servers connected to that subnetwork. The router function extends access to servers connected to another subnetwork at another access point of that IED which hosts the router function. However, a router restricts the access to those services which use a networking layer, all layer 2 services such as basic GOOSE and sampled value messages are not allowed to cross it.

Clock — a master clock at this subnetwork, which is used to synchronize the internal clocks of all (other) IEDs connected to this subnetwork.

Routers and clocks are connected to a Subnetwork via their access points.

Observe that the communication addresses defined for the access points within a subnetwork are access point addresses for building associations. The rules to derive communication addresses of the server internal elements are defined within the protocol mappings on the base of the IED data model as defined in IEC 61850-7-x, e.g. within IEC 61850-8-1 for the MMS mapping.

6.5 Modelling of redundancy

Redundancy can be introduced to enhance the safety or availability of a system, and at different levels of the system:

- IED internal: this is beyond the scope of the IEC 61850 series, and therefore not describable with SCL. It is hidden in the IED HW/SW and externally visible just by error messages if something has failed. IED specific DATA might have to be introduced for these error indications.
- Communication system level: If the communication system is doubled, but below the addressing level provided for a logical access point, this can be described in SCL at the level of physical connections of an access point. There might be additional SCSM specific parameters or application level supervision data, if the redundancy issue is taken up in the stack mapping. Other communication system redundancy mechanisms can only be described at physical level. A typical example is an Ethernet ring based on switches. It provides redundancy against the failure of one switch in the ring, it is however normally not seen within an SCD file. However SCL provides some optional means to describe the physical connections at port / cable level also for rings.
- Application level: this shall be modelled in SCL. A typical example is the main 1 and main 2 protection IED. Each IED instance providing application redundancy is explicitly modelled having its own name, and all explicitly provided additional communication subnetworks are also modelled in the SCD file as indicated in Figure 7. Any coordination between redundant functions is done between the logical nodes which implement the function.

6.6 Data flow modelling

Conceptually the IEC 61850 data flow has logical nodes on servers or publishers as source, and logical nodes as clients respectively subscribers. The real connections/associations however are built at communication profile level (e.g. MMS/TCP), and these 'association channels' can be assigned to a client/subscriber IED, an access point, a logical device or – as in the IEC 61850 base model – a logical node. If the channel is built by an IED, then all LNs hosted in this IED can use it in their client role. It should be observed that these channels / associations are also the level of granularity on which access rights are checked, i.e. each association is seen as one user respective client with a certain role.

SCL allows the data flow to be modelled at two levels. At the channel/association level the GOOSE or SMV subscribers are whole IEDs, respective the IED access point connected to the same SubNetwork as the server, while report clients are LN instances, such as in the original IEC 61850 client model. If in this case several client LNs share the same channel, it is recommended taking the LLN0 as the client LN, because LLN0 represents a whole logical device.

At data object level the data flow is modeled by a list of signals which shall be fed into (are input data of) a logical node. This can be modeled purely on an SCL level, or, if the IED supports this, even on the IEDs LN data model by means of data objects of CDC ORG (see IEC 61850-7-4). Also here it is often the case that the same incoming data object shall be used by several logical node instances. In this case it is also recommended to map the input data into the LLN0 instead of mapping it twice to two different LN instances.

One of the big advantages of IEC 61850 is that the communication-related data flow is defined on top, but independent from the application level data model. To make it easier for an engineer to understand and define this data flow, the SCL language restricts the definitions in 7-2 as follows: data set definitions referenced by a control block must be in the same logical node as the control block. This means automatically, that all GOOSE and SMV data flow definitions are in LLN0. It is recommended, if the IEDs allow this, to also keep report data flow definitions there. Any online changes not following this convention cannot be documented in SCL language.

7 SCL description file types

SCL files are used to exchange the configuration data between different tools (conceptual tool roles as defined in Clause 5), possibly implemented by different manufacturers. As already mentioned in 5.1 (see also Figure 1), there are at least six different purposes for SCL data exchange, and therefore six kinds of SCL files to be distinguished for the data exchange between tools. This is done by means of different file extensions. Nevertheless, the contents of each file shall obey the rules of the System Configuration description Language (SCL) defined in the next clause. Each file should contain a version and revision number to distinguish different versions of the same file. This means that each tool has to keep the version and revision number information of the last file exported, or read back the last existing file to find out its version.

NOTE The version identifies versions of the SCL file, not versions of the data models used within the tools. This is defined in IEC 61850-7-3, IEC 61850-7-4 or is a private issue of the tools.

The following types of SCL files are distinguished:

- Data exchange from the IED configurator to the system configurator (corresponding to items b) and c) of 5.1). This file describes the functional and engineering capabilities of an IED type. It shall contain exactly one IED section for the IED type whose capabilities are described. The IED name shall be **TEMPLATE**. Furthermore, the file shall contain the needed data type templates inclusive logical node type definitions, and may contain an optional process, line or substation section, where the highest-level name shall be **TEMPLATE**. When importing the file into an SCT, the hierarchy of elements named TEMPLATE is used to identify the first named element to be instantiated in the project, and all elements with a name different than TEMPLATE are considered to be instantiable, based on the name. If a process TEMPLATE is defined, the binding of logical node instances to primary equipment indicates a predefined functionality. Any process in which this IED shall be used must match an appropriate process topology part (example: a CSWI LN bound to an equipment of type CBR is only allowed to control a circuit breaker; a CILO bound to a line disconnector implements the interlocking logic for a line disconnector). There might be an optional Communication section defining possible default addresses of the IED. A specific SCSM might make this mandatory for some address parts.

  The file extension shall be ICD for IED Capability Description.

- Data exchange from the IED configurator to the system configurator for a single IED preconfigured specifically for a project, e.g. to include a preconfigured instance file or IED instance value changes or data model modifications. In this case the IED has its project specific name, it may also have project specific addresses and a data model possibly included with some data set definitions preconfigured for the project. There might exist already a binding of IED LNs to the project specific single line diagram. This type of IED SCL file is typical for IEDs whose number of LN instances depends on the project specific single line diagram or on other IEDs available in the system, or it is used during IED modification process. It may contain a data set and control block definitions, which must either be identical to those in the system tool in case modifications are transferred after system engineering, or, in case of a first instantiation of this IED, can be taken as default or as preconfigured data. It may contain input sections without the referenced DATA sources. These shall be identical to that from a previously imported SCD file, however links to internal signals (intAddr values) may be added.

  The file extension shall be IID for Instantiated IED Description.

- Data exchange from a system specification tool to the system configurator. This file describes the single line diagram and functions of the substation and the required logical nodes. It shall contain a process section which may be composed of Substation, Process and/or Line elements, and may contain the needed data type templates and logical node type definitions. If logical nodes allocated to the Substation section are not already allocated to an IED, the IED name reference (value of *iedName* attribute of the *LNode* element) shall be **None**. If an LN in the substation section is not bound to an IED and also has no logical node type defined, then only the mandatory part of this LN according to IEC 61850-7-4 is specified. If part of the SA system is already known, this might optionally be contained in IED and Communication sections.

  The file extension shall be SSD for System Specification Description.

- Data exchange from the system configurator to IED configurators (corresponding to items d) and e) of 5.1). This file contains all IEDs including the configured data flow and needed DataTypeTemplates, a communication configuration section and a substation description section.

  The file extension shall be SCD for System Configuration Description.

- Data exchange from the IED configurator only to the IED. It describes the communication-related part of an instantiated IED within a project. The communication section contains the address of the IED. The substation section related to this IED may be present and then shall have name values assigned according to the project specific names. It is a vendor specifc SCD file, possibly stripped down to what the concerned IED shall know (restricted vendor specific view of IEDs). If a compression method is applied, those according to RFC 1952 is preferred. Observe that in the general case more information than this has to be loaded onto an IED to have it completely configured, e.g. relation of internal signals to HW terminals, programs in the form of IEC 61131-3 or other code, or local control panel configuration information.

The file extension for the SCL part (if any) shall be .CID for Configured IED Description.

- Data exchange between system configurators of different projects. This file describes the interfaces of one project to be used by the other project, and at reimport the additionally engineered interface connections between the projects. It is a subset of a SCD file, containing the interfacing parts of the IEDs to which connections between the projects shall be engineered, and fix IEDs referenced by them to not lose the source object of already defined references. Therefore additionally to an SCD file it states at each IED the engineering rights and the owning project from the view of the using (importing) project.

  The file extension shall be SED for System Exchange Description.

A more formal definition of most restrictions for the given parts is given in the XML schema syntax in Annex E. Observe however, that this formal definition is informative only and does not belong to the normative SCL language definition.Observe further that not all restrictions e.g. those on IED name and Substation name mentioned above can be described in the schema. To understand the used schema elements, refer to Clauses 8 and 9.

An IED which is claimed to implement a server / publisher or client / subscriber according to the IEC 61850 standard shall be accompanied by an ICD file, respectively by a tool capable of generating an ICD file, or a project specific IID file, respectively a tool capable of generating a project specific IID file for this IED, and shall be able to consume an SCD file or be accompanied by a tool which can consume the SCD file to configure the communication part of the IED from this SCD file, within the limits declared in the ICD file or the IID file produced previously by the IED tool .

It shall be kept in mind that, for very flexible IED types, there might exist several ICD files. In this case the manufacturers IED type can be seen as an IED class similar to logical node classes in IEC 61850-7-4, which allows a lot of functionality to run on the IED hardware, however not all at once. Each ICD file then is a runnable (implementable) subset of all possibilities of the IED class. Only where all available functions and function instances can run on the IED hardware, will there exist only one ICD file. This issue is illustrated in Figure 8 for the most general case.

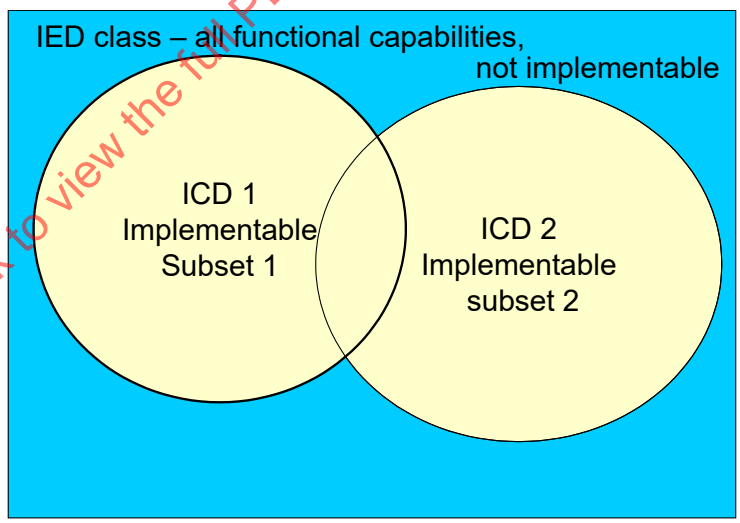


Figure 8 – ICD files describing implementable IED types of a general IED class

8 SCL language

8.1 Specification method

The SCL language is based on XML (see Clause 2).

The syntax definition is described as a W3C XML schema. The remaining clauses define the appropriate XML schema for SCL and explain its usage in text, enhanced by appropriate (incomplete) examples illustrating the use of the specific features defined, and by additional written requirements, restrictions, and relations to the object model, which shall be used or checked by the application reading or building an SCL file. The complete normative XML schema definition is contained in the SCL code component defined in clause 1.3. It also contains the formal definitions of those constraints which are easily formulated in a XML schema. Constraints on the object model which are not or not easily able to be formulated in XML schema are additionally described in the appropriate clauses.

To keep the syntax compact and extensible, the type feature of XML schema is used where appropriate. This introduces a schema element inheritance structure. The inheritance structure of the main SCL elements is shown in Figure 9 as a UML diagram.

UML diagrams can also show containment relations between SCL elements. It has to be kept in mind that these relations are relations between the SCL language elements, and not between the objects represented by the elements, which are shown in Figure 6. However, it has been attempted to keep the XML element relations as close to the object relations as possible.

The following naming conventions are used within the schema:

- schema type names start with the small letter t (for example *tSubstation*);
- attribute group definitions start with the acronym ag (for example *agAuthorization*);
- attribute names start with a small (lower case) letter (for example *name*);
- element names start with a capital (upper case) letter (for example *Substation*).

Nearly all SCL elements are derived from the *tBaseElement* base type, which allows adding *Private* sections and a descriptive *Text* to the element. It also allows adding additional sub-elements and attributes from other namespaces (other than the target namespace http://www.iec.ch/61850/2003/SCL) – such elements must however appear first among all sub-elements. This allows for easy (private) extensions of the model. An example can be found in Annex C.

The next level of element types is based on *tBaseElement*:

- *tUnNaming* adds an optional description attribute *desc;*
- *tNaming* adds the optional description attribute *desc* and a mandatory name attribute *name;*
- *tIDNaming* adds the description attribute *desc* and a mandatory identifier attribute *id*.

In all the previous types, *desc* is a XML normalizedString, i.e., a string that does not contain any carriage return, line feed, or tab character. Its default value is the empty string. Attribute *name* is of type *tName*, i.e. also strings that do not contain any carriage return, line feed, or tab character, but cannot be empty. Attribute *id* of type *tID* is more restrictive than *tName*: any whitespace character is not accepted and size is limited to 255 characters.

The resulting inheritance relations for the power system-related objects is shown in the UML diagram of Figure 15. Due to this inheritance, also of attributes or of attribute groups, not all attributes are directly defined at an element definition. Nevertheless the description in the following clauses also describe the inherited attributes, possibly with a reference to a previous description.

For better segmentation and re-use, the whole SCL schema is split into several files containing type definitions (see Table 1).

Table 1 – The files composing the XML schema definition for SCL

File name	Description
SCL_Enums.xsd	The used XML schema enumerations
SCL_BaseSimpleTypes.xsd	The basic simple types used by the other parts
SCL_BaseTypes.xsd	The basic complex type definitions used by the other parts
SCL_Substation.xsd	The process (Process, Line, Substation) related syntax definitions
SCL_Communication.xsd	The Communication-related syntax definitions
SCL_IED.xsd	The IED-related syntax definitions
SCL_DataTypeTemplates.xsd	The data type template-related syntax definitions
SCL.xsd	The main SCL schema syntax definition, which defines the root element of each SCL file

In the following schema definition clauses it is assumed that the SCL schema definition file starts as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.iec.ch/61850/2003/SCL"
  xmlns:scl="http://www.iec.ch/61850/2003/SCL"
  xmlns="http://www.iec.ch/61850/2003/SCL"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified"
    finalDefault="extension" version="9999A9">
```

where version 9999A9 is the year, index letter and release number stating the SCL schema version, which is 2007B4 for this document. The schema then ends with

```
</xs:schema>
```

This schema part is not repeated in the following clauses and subclauses. For a complete schema definition containing the contents of all above files, see the SCL code component defined in clause 1.3.

The UML diagram given in Figure 9 gives an overview of how the SCL schema is structured.

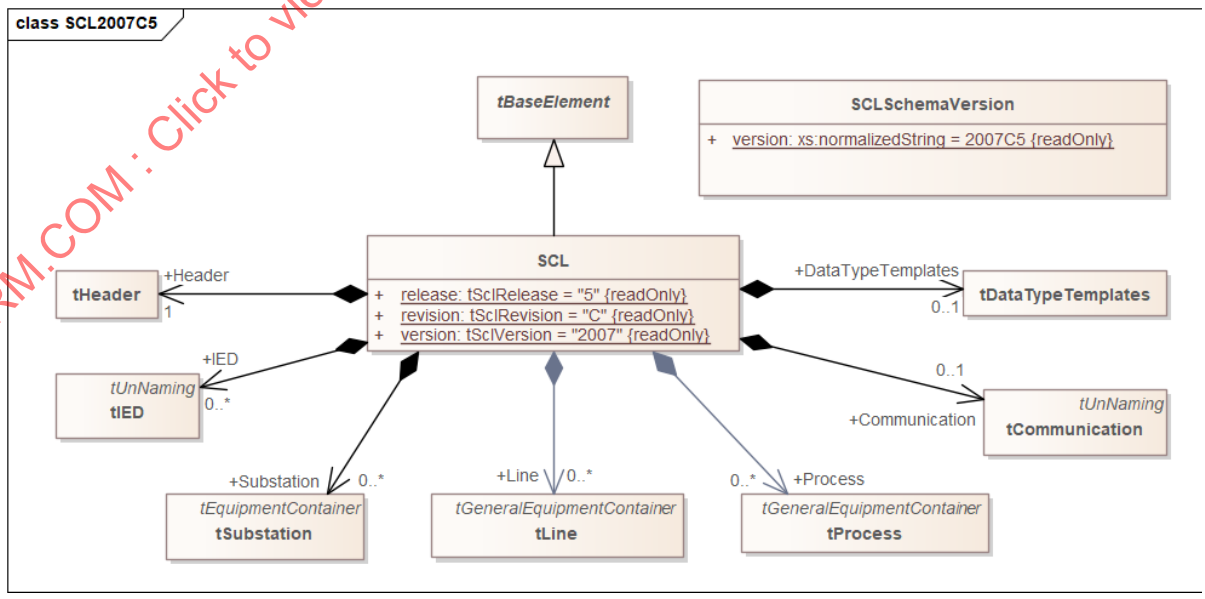


Figure 9 – UML diagram overview of SCL schema

The basic SCL element is derived from a *tBaseElement* schema type, which allows to contain for example Private and Text definitions. Furthermore, the SCL element shall contain one Header element of type *tHeader*, and may contain a process description with one of the elements process of type tProcess, Line of type tLine, and Substation elements of type *tSubstation*, a Communication section of type *tCommunication,* one or more IED elements of type *tIED*, and a DataTypeTemplates section of type *tDataTypeTemplates*. All these element types are handled in later clauses; the usage of Line and Process elements might be detailed in other parts of the standard.

In some cases, the data format of values is important. Wherever possible, the schema defines the data type and therefore also its coding (lexical presentation). But even in cases where this is not possible, the data type coding of XML Schema shall be used. If not explicitly expressed, all element values are XML Schema *string*s, and all attribute values are of the XML schema type *normalizedString*, i.e. they are not allowed to contain tab, carriage return and line feed characters. SCL may use values to identify another element of the XML file. When doing this, the referenced element shall exist in the file, otherwise the structure is invalid. For example (full definition of this example is given later) in element GSE, a specific attribute called cbName indicates the element GSEControl and attribute IdInst indicates the Ldevice containing this GSEControl, for which the GSE has been created, and if this GSEControl is not present, the GSE shall not exist. Further restrictions may be stated either in this part of IEC 61850 or in other parts of the IEC 61850 series, mostly IEC 61850-7-x, IEC 61850-8-x and IEC 61850-9-2. If any XML schema data type is used, it is referenced with the prefix *xs:*, for example *xs:decimal* for decimal number coding. For convenience, an overview about coding of the most types used in SCL is given in Table 45.

8.2 Language versions and compatibility

There are always some reasons why a defined language has to be changed, which leads to different language versions.

- Enhancements: adding new features; this has to be done to support new functionality, and leads to a new version indicated by the year of appearance, for this version it is 2007. To keep compatibility, the following enhancement rules have to be observed also for future compatible SCL versions. If for some reason they can no longer be observed, then a new, incompatible SCL name space has to be defined.
  - Adding of new optional attributes is allowed. If they need a value, they shall have default values, whose meaning is as far as possible identical to the missing of the attribute in older versions.
  - Adding of new elements is allowed at the end of existing type definitions.
  - To allow forward compatibility, any new element, whose understanding is essential for communication interoperability, must be marked with the *mustUnderstand* attribute (see later).
- Fixing errors; this is necessary if there exist faults or inconsistencies, or if interoperability problems arise due to unclear or wrong wording in the description or specification. This reason is mandatory and must be performed, even if it endangers compatibility. However, if there are choices, it should be done in the most backward compatible way. This leads to a schema revision, indicated by a revision index (letter) starting with A.

The following language changes are forbidden for a compatible language name space, because they lead to compatibility problems:

- Removing of old features. For backward compatibility they are still allowed in older language instances, but the usage in newer versions is deprecated. The deprecation is normally indicated by removing the feature from the schema of the new version. Nevertheless it is allowed in language instances coming from older versions, and must be accepted by a receiver.
- Changing of existing features, especially their semantics; this endangers compatibility, therefore it is forbidden. Instead 'old' features are deprecated (see above), and new ones added, which then replace the deprecated features.

- Changing of existing default values. This allows a receiver or processor to use default values of newer versions also for older language instances.

There is a clear separation between mandatory fixing of errors, which might lead to incompatibilities, and adding enhancements, which are done in a compatible way. To allow backward compatibility as well as forward compatibility, the *may ignore* and *must understand* rules are introduced into the language definition. These allow having different language versions for the same SCL language name space.

8.2.1 MustUnderstand rules

The MustUnderstand / MayIgnore rules have to be followed when reading an SCL file. When producing SCL output, this should conform to the claimed supported SCL version/revision/release.

Elements, which a tool or an IED must understand to produce interoperable results, shall be declared as *mustUnderstand* and marked with the *mustUnderstand* attribute with value *true*, so that the tool processing the instance knows if it can ignore the element or not. All elements which the tool does not understand and which do not have the *mustUnderstand* property, can safely be ignored. The 'may ignore all' strategy for elements (tags) is taken, i.e. ignore the element and all its contained contents.If a known element contains directly below it an element with mustUnderstand property which a tool does not understand, then it must also ignore the known containing element.

For attributes just the attribute not understood is ignored. This means especially that there is no 'mustUnderstand' possibility for attributes, only for elements. Therefore adding of attributes to the language is done only as optional attributes with a defined default value in the newer version, which is backward compatible to 'not knowing this attribute'. For later compatibility it is good practice to use these default values from the schema by not explicitly writing them into the SCL instance. This is possible because once released default values are not changed in the schema, as long as the attribute itself is needed.

Observe that if attributes need to be understood, then a new element with *mustUnderstand* property holding these attributes can be introduced.

It is important to see that whether a tool can ignore something or not is also dependent on the purpose of the tool. When defining 'mustUnderstand' explicitly in SCL, then this always refers to the system configurator and the IED configurator as defined in Clause 5 for the purpose of interoperable communication. Other applications for which SCL may be used can have other demands on 'mustUnderstand'.

Observe that although not formally defined, the *mustUnderstand* property is practically true for all defined elements from the 2003 SCL version in the *Communication* section, *IED* section (with exception of the IED capability element) and *DataTypeTemplate* section. The elements of the *Substation* section might need 'mustUnderstand' quality only for specification tools and application configuration tools, which are not mandatory and outside the scope of this part of IEC 61850.

From the meaning / scope, *mustUnderstand* always refers to the parent element. If a parent element (e.g. IED) has no mustUnderstand property, then it may be ignored by tools which do not need to know about it (e.g. about IEDs). However, if they have to know about IEDs, they must understand all elements within the IED element, which have mustUnderstand property, e.g. the *AccessPoint* element. In general, if an element is marked as *mustUnderstand*, then any tool which does not understand the element is not allowed to use the (known) parent element. The following example illustrates the use of the *mustUnderstand* feature.

An IED defines a GOOSE control block as follows:

```
<GSEControl name="GoCB02" appID="GoCB02" datSet="CmdResv">
    <IEDName>AA1_D1_Q11A1</IEDName>
    <IEDName>AA1_D1_Q10A1</IEDName>
    <IEDName>AA1_D1_Q07A1</IEDName>
    <Protocol mustUnderstand="true">R-GOOSE</Protocol>
</GSEControl>
```

If the tool importing this does not understand the *Protocol* element, then it is also not allowed to use the *GSEControl* element containing it (e.g. add IED names), although *GSEControl* is known to it.

Tools should in any such case give a warning to the user.

8.2.2 SCL name space and versions

For all compatible versions of SCL the same name space as defined in 8.3.5 is kept.

For concrete verification of correct generation of an SCL instance according to its assigned version, the following is introduced.

The SCL element tag has a *version* attribute, which for backward compatibility is in general optional with default value 2003, and for instances assigned to the here defined version of SCL required with value 2007. This attribute indicates the SCL (schema) version according to which the SCL instance has been produced by means of the year of the released IS, i.e. its assigned version. Additionally, any error fixing revisions within each version are indicated by the *revision* attribute, starting with A for the first released version revision. The version value for this version of SCL shall be 2007, and the revision value B. If error-correcting corrigenda of this standard follow before any new version of this standard is published, the first corrigendum will get the identification B, the next C, etc. Fixing of interoperability related errors in between revisions is indicated by a release number. The first released revision will always be release 1. The release belonging to this standard is 3.

The special XML schema for this current edition of IEC 61850-6 is contained in the SCL code component defined in clause 1.3, and shall be used for all tests on SCL instances which claim to be produced according to this SCL version. From this follows automatically, that the SCL version attribute is mandatory required for all SCL instances containing elements or attributes introduced after 2003.

For backward compatibility all tools have to accept SCL instances from older as well as newer versions, including the features deprecated in the newer version(s) as far back as declared with the tool version. Therefore a tool input cannot be verified against the schema of the version defined in this standard, but at best against the schema version with which the input instance is produced. The general schema in Annex E gives a hint as to what shall be tolerated by a tool supporting the valid 2003 version as well as this current version.

Naturally 'old' tools processing SCL instances from newer versions can not handle what they can not understand. The only problem which might arise here is if the SCL instance contains new elements with a 'mustUnderstand' property, which is not known to tools/IEDs according to the first SCL version. Every tool/IED after this first version shall use the mustUnderstand property to decide if it can safely ignore an element, which is not understood, or if it has to stop processing with an appropriate error message. It is recommended to upgrade 'old' tools at least so that they can follow the mustUnderstand rule, or to implement the upgrading / downgrading rules from Annex I.

Further, tools understanding the new version should also read 'old' instances, if they are produced according to the rules defined here.

The SCL language version as defined here is related to the SCL schema version as defined in Subclause 8.1. However, not all schema versions will be released, and SCL language versions, even bug fix versions, will only be defined for released / published SCL schema versions.

8.2.3 Incompatibilities to earlier versions

This current version of SCL with version identification 2007 is backward compatible to all previous versions with the following exceptions.

- The authentication code 'week' has been corrected to 'weak' (error correction).
- The Private element's *type* attribute is required (error correction).
- The *sampleSynchronized* attribute of SMV options (agSmvOpts) is no longer allowed to be false (error correction in 9-2).
- The attribute value *FuncName* of the *nameStructure* attribute in the *Header* element is no longer supported. The *nameStructure* attribute shall be ignored by tools. Systems working with the 2003 functional naming must be modified appropriately to stay compatible, by either changing to IED based naming only, or allowing the use of the *IdName* attribute of *LDevice* (i.e. full change to this version of SCL).
- The introduction of the *mustUnderstand* attribute; it is currently used for the *Protocol* element contained in GOOSE and SV control block definitions.
- The order where the Log element appears has changed: it shall appear directly before any control block definitions which belong only to LN0 (like GOOSE control blocks).
- If the log control block does not reside in the same logical device as the log, the *ldInst* attribute shall be stated explicitly.
- The access point name allows only alphanumeric characters and underscore (_). Its length is restricted to 32 characters.
- The newly introduced LDevice attribute IdName leads to incompatibilities with edition 1 (2003A) tools, as ignoring this attribute leads to wrong communication level configuration.
- The attribute ReportControl.rptId shall no longer have the empty string value.
- It is clarified that the meaning of the *maxAttributes* attribute of the *ConfDataSet* element denotes FCDAs and not basic attributes.
- The modeling of the transformer neutral point has been corrected
- The *max* attribute of the *SupSubscription* element has been replaced by two special attributes for GOOSE and SV supervision.
- The length of *DataTypeTemplate* identifiers is restricted to 255 characters.
- *EqFunction* and *EqSubFunction* are used below Equipment elements instead of Function / SubFunction
- The length of Enumeration strings is restricted to 127 characters of ISO Latin and Latin-1 supplement characters
- The SubNetwork type code is mandatory and not the empty string, except in SSD files.
- The value of RptEnabled.max, if defined, shall be >0.
- The meaning of missing originalSclVersion / originalSclRevision attributes is SCL version/revision 2003A
- The string 'None' is forbidden as an IED name. It is only allowed as an IED name substitute inside the Process / Substation section to specify logical nodes without relation to an IED.

It is recommended to fix these issues together with the implementation of the mustUnderstand property also in 'old' tools. Then they can handle all future SCL versions. For compatibility between versions see also Annex I.

8.3 SCL language extensions

8.3.1 General

The SCL language elements without those serving extension purposes are designed for a specific purpose as described in Clause 5. It can however be used with smaller or bigger extensions such as additional attributes for additional (engineering) tasks. Furthermore, it leaves some communication stack-dependent definitions to the SCSMs. Therefore, 8.3.2 to 8.3.7 describe SCL extension possibilities.

8.3.2 Data model extensions

Extensions of the data model with semantically new LNs and DOs are covered by the rules stated in IEC 61850-7-x for extensions, and by the SCL approach as a meta language to the data model, i.e. data model element identifications do not appear in the language syntax itself. The name scope of logical node classes, data objects and CDC attributes are described in SCL by stating the appropriate name space values within the appropriate DATA attributes. If additional base data types are needed, then this has to be defined as a schema extension.

8.3.3 Additional semantics to existing syntax elements

Some language elements of SCL such as *desc* and *Text* have a weakly defined semantic, which can be extended by some application. Some elements such as the parameter element *P* have been left open on purpose. An SCSM shall define (additional) semantics to these elements. This is done by defining a *type* value for a P parameter with an own semantic.

8.3.4 Data type constraints

The usage of XML schema based data types on the syntactic level already allows the further restriction of the range of some values. A restriction shall use one of the allowed subtypes of the types defined in this core language.

8.3.5 XML name spaces

For all tag elements inheriting from tBaseElement, (sub-)tags and attributes can be added. These shall however belong to a defined XML name space with defined semantics for all these elements. It is recommended to define the used name spaces at the main tag (SCL), although standard XML allows to add it also below at elements from this name space. Observe that standard XML rules allow to replace this definition.

This namespace shall not be the same as the target namespace of the SCL schema (see below). For private name spaces, the used internal name space abbreviation should start with the character **e**. IED configurator tools shall be aware that the used name space abbreviation might be changed by the system configurator, if its relation to the referenced URI is not unique across all used ICD/IID files. An example of a standard extension for single line or communication diagram layouts is given in Annex C. The name space URI of this version of the SCL, which shall be used as default name space in all SCL files, is:

```
xmlns:scl="http://www.iec.ch/61850/2003/SCL"
```

All tools, which comply with this part of IEC 61850, shall be able to import an SCL file with name space definitions, and at least interpret the SCL elements of the default name space. Therefore any SCL file shall have the SCL name space as default name space:

```
xmlns="http://www.iec.ch/61850/2003/SCL"
```

Name spaces other than the SCL core, which are not understood by the tool, shall be ignored by it. This especially means that an IED tool which exports data of its own XML name space to an ICD file, can not expect that this information is contained, respectively preserved, in a SCD file coming from the system configurator tool or another manufacturer's IED tool, if it is not contained within a *Private* section.

Other parts of IEC 61850 also define extensions of the SCL using specific XML Namespace, stored in a dedicated *Private* section. This is the case for technical specifications (e.g. IEC 61850-80-1 for mapping to IEC 60870-5-101 or -104) or technical reports (e.g. IEC 61850-90-30 for SCL function modelling). In the case of technical reports, the XML extension introducesan experimental new feature which may be included later in the SCL namespace itself. In this context, the extension may contain references to SCL elements. In the case where users want to use this extension, they will have to be aware that all tools implied in the engineering have to support this extension to maintain the SCL references (e.g. when an extended element references a Bay, when the Bay name is updated, all references shall be updated even within the private extensions introduced by technical reports).

NOTE 1 The SCL schema is built in such a way that if the private namespaces are specified in the header but the corresponding schemas are unknown, an XML validator is still able to correctly validate the file (for the parts that are not defined in the SCL schema, the validator will typically only check that they are well-formed).

NOTE 2 The SCL schema demands that elements from private name spaces appear in an SCL file before the elements defined in the SCL schema.

8.3.6 Private data

For small extensions either by a manufacturer or for a specific project the *Private* elements can be used. The advantage of private elements is that the data content is preserved at data exchange between tools.

Private data entities appear on several levels of the SCL. The contents of these XML elements is, as seen from the SCL, transparent text. If the private part contains XML data, then this has to use an explicit name space, which cannot be the SCL name space. The Private element allows also to reference other files by means of a URL at its *source* attribute.

The handling within tools shall be as follows:

The private data is owned by a tool respective by a tool category (for example, a picture generator). The owner is allowed to modify its contents, and normally is the only one able to interpret the data. All other tools, which read private data, have to preserve (store) its contents on SCL import, and regenerate it at the same place if an SCL file containing this part is produced/exported.

Private data for different purposes shall be distinguished by the value of its *type* attribute. If manufacturers use a Private definition, this type attribute value should start with a manufacturer-specific string part.

The Private elements have the schema type tPrivate, which is defined as follows:

```
<xs:complexType name="tPrivate" mixed="true">
    <xs:annotation>
        <xs:documentation xml:lang="en"> Allows an unrestricted mixture of character content, element content and attributes from any namespace other than the target namespace, along with a mandatory type attribute. </xs:documentation>
    </xs:annotation>
    <xs:complexContent mixed="true">
        <xs:extension base="tAnyContentFromOtherNamespace">
            <xs:attribute name="type" type="xs:normalizedString" use="required"/>
            <xs:attribute name="source" type="xs:anyURI" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes of the Private element are defined in Table 2.

Table 2 – Attributes of the Private element

Attribute name	Meaning, usage
type	Distinguishes different (private) purposes of the element contents. The manufacturer or tool name shall be included into the type to be sure it is unique. The type attribute is required in order to know who shall process this part.
source	URL to some file, which contains the private information; only the URL is preserved by the processing tool, not its contents (this stays where it is and has to be preserved with means outside the tool responsibility).

Private data can be contained within the *Private* element, or in an external file referenced via the *source* attribute of the *Private* element. As a rule of thumb this second option shall always be used if the amount of private data gets big in relation to the standardized part, e.g. above 1-2 kB.

NOTE Due to the engineering process as described in Clause 5 IED configurators are allowed to put Private elements into the IED section, the related ConnectedAP section and the DataTypeTemplate section for types used by the IED.

8.3.7 Another XML syntax

A completely new standardized or private XML-based syntax for another XML file may be used to extend the SCL data model with additional objects or attributes. In this case, references to the objects contained in the SCL model shall be defined in this new XML file, and the naming philosophy of this part of IEC 61850 shall be followed to be able to identify the objects. The *source* attribute of a *Private* element can be used to link to such additional XML files.

8.3.8 Summary: Standard conformance for extension handling

A tool claiming conformance with this part of IEC 61850 shall as a minimum handle any extensions as follows:

- import and export the SCL language elements as a default XML name space; understand all parts of the syntax referring to the capabilities of the handled IEDs and the intended functionality of the tool; ignore all SCL language elements which it does not understand, following the mayIgnore / mustUnderstand principle (see 8.2.1);
- keep all data in private sections and all text elements from import to export (except if modified on purpose within the tool). Keep all data of IEDs, which are not handled, if an SCD file is exported;
- accept syntactically correct XML name space extensions on import without error message, even if the corresponding contents are ignored.

8.3.9 Extension example

The following extract of an SCL file shows how extensions based on private XML name space can be used for additional XML attributes, additional elements, and for XML elements within the data part of a *Private* element.

```
<?xml version="1.0"?>

<!-- Augmented example file with:

  – Private element

  – using extensions from other namespaces

-->

<SCL xmlns="http://www.iec.ch/61850/2003/SCL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
        xsi:schemaLocation="http://www.iec.ch/61850/2003/SCL  SCL.xsd" xmlns:ext="http://www.private.org">

    <Header id="SCL Example T1-1" nameStructure="IEDName"/>

    <Substation name="baden220_132" ext:myAttribute="my extension attribute">

        <ext:MyElement>This is my extension element – can be removed if not understood</ext:MyElement>

        <Private type="mytype" ext:hello="bla bla">This is my private element <ext:dummy>with sub-
elements</ext:dummy> and a privately defined attribute; must be reproduced at output</Private>

        <PowerTransformer name="T1" type="PTR">
```

Observe that all elements (above the *MyElement*) from other name spaces (*ext* above) other than the default SCL name space must come before any SCL elements.

8.4 General structure

An SCL – XML document starts with the XML *prolog,* and then continues with elements as defined later. The *prolog* shall contain the identification of the XML version and the character coding used. UTF-8 coding is the preferred coding and shall be supported supported if standard conformance is claimed. The XML encoding attribute shall be processed in a case-insensitive way. SCL generators shall always use upper case encoding.

The whole SCL definition part is contained in the SCL element:

```
<?xml version="1.0" encoding="UTF-8"?>

<SCL xmlns="http://www.iec.ch/61850/2003/SCL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.iec.ch/61850/2003/SCL SCL.xsd" version="2007" revision="C" release="5">


<!-- here come the Header/Substation/IED/Communication/DataTypeTemplate sections as defined in Clause 9 -->

</SCL>
```

where **SCL.xsd** gives the concrete file containing the SCL schema definition.

For an XML processor, this example assumes that the SCL schema definition (i.e., the files enumerated in Table 1) is in the same directory as the SCL instance file. If this is not the case, the full path to the schema must be given here. Alternatively, most XML processors allow you to provide the location of the schemas manually (outside the instance document). Anyhow, the *xsi:schemaLocation* attribute is only needed if syntax verification against a specific schema needs to be carried out.

The SCL element shall contain a header section, and at least one of the following sections: Process or Substation, Communication, IED, DataTypeTemplates, which are further explained below. The Process/Substation and the IED sections may appear more than once. Figure 9 gives an overview as an UML diagram.

Most elements are derived from the tBaseElement type, and therefore inherit the options to contain *Text* and *Private* elements as well as the capability to contain elements and attributes from other name spaces. The elements derived from its sub-types *tUnNaming*, *tNaming*, and *tIDNaming* additionally inherit the *desc* attribute.

All SCL level references to objects on an IED use the IED-related names, i.e. the IED name and LD instance name, even if at communication level other identifications might be used. This is valid for references from the substation section to logical nodes on the IED, but also for references within an IED, e.g. to define data objects which are the members of data sets.

Observe further that the SCL element has the attributes *version* with value 2007 for this version of the SCL language, *revision* with value B for this revision of the 2007 language version, and *release* with value 3 for this published release.

8.5 Object and signal designation

8.5.1 General

The SCL model allows two kinds of object designation:

1) a technical key, which is used on engineering drawings and for signal identifications. This is contained in the attribute *name* as identification of each object. If this value is used as reference to an object, it is contained in an attribute name starting with a string denoting the reference target object type, and ending with the string "*Name*", e.g. iedName as reference to an IED. The technical key is used within SCL for referencing other objects. Observe that *name* is a relative identification within a hierarchy of objects;
2) a user oriented textual designation. This is contained in attribute *desc*. Attributes are not allowed to contain carriage return, line feed or tab characters. The semantics of *desc* shall also be relative within an object hierarchy.

Furthermore, a general description tag *Text* can be used to add descriptive textual data. The meaning of this data is on purpose not specified further. Each tool shall preserve imported text data for export.

8.5.2 Object designations in an object hierarchy

In case of the hierarchically structured objects of the substation structure and the product structure, both *name* and *desc* attributes for each object contain only that part which identifies the object within this level of the hierarchy. The full object reference is a pathname and consists of the concatenation of all name parts of higher hierarchy levels up to this level. It is up to the configuring engineer to ensure that the references are unique after concatenation. This shall be reached by using a designation (syntax) convention as specified in IEC 81346-1. This especially means that names of all levels can be directly concatenated to a path name, if the higher level name ends with a number and the lower level name starts with an alpha character or else an intervening character, preferably a dot (.), shall be put between them. Other separation characters may be specified for name mapping in SCSMs or according to IEC 81346-1. Beneath the mandatory usage of IEC 81346-1 for name syntax, it is strongly recommended to use the whole IEC 81346 series for the derivation of functional and IED product names as technical keys. In this case, it should be observed that the special IEC 81346 separator characters like =, +, – shall not appear within SCL names. Only the dot (.) is allowed if names are substructured.

Transition objects, i.e. objects appearing in more than one hierarchical structure, may be identified by several references, one in each structure. In the case of SCL, this applies especially to logical nodes, which are found in the substation functional structure as well as in the IED product structure. There might be other transition points between different structures, but their modelling is outside the scope of SCL.

The designation of objects which are also intended to be displayed in any kind of user interface can have a list of labels by the mean of the element Labels which will contain a list of Label elements which are textual elements with a *lang* attribute allowing to identify the language of the label, allowing translation, and also an optional *id* attribute allowing to create different labels for different purpose for the same element. The language attribute is following the RFC 1766 Language Tags (i.e. it can be "en", "en-US", "fr", "fr-FR", …).

Here is the XML schema definition of labels:

```
<xs:complexType name="tLabels">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Label" type="tLabel" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="tLabel" mixed="true">
    <xs:simpleContent>
        <xs:extension base="xs:normalizedString">
            <xs:attribute name="id" type="tID" use="optional"/>
            <xs:attribute name="lang" type="xs:language" use="required"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

8.5.3 Signal identifications to be used in the communication system

According to IEC 61850-7-2, signal identifications are built from the following parts (see Figure 10):

a) a user defined part identifying the logical device LD in the process (LDName);

b) a (function-related) part to distinguish several LNs of the same class within the same IED/LD (LN-Prefix);

c) the standardized LN class name and the LN instance number, which distinguishes several LNs of the same class and prefix within the same IED/LD;

d) a signal identification inside a LN consisting of data and attribute name as defined in IEC 61850-7-3 and IEC 61850-7-4.

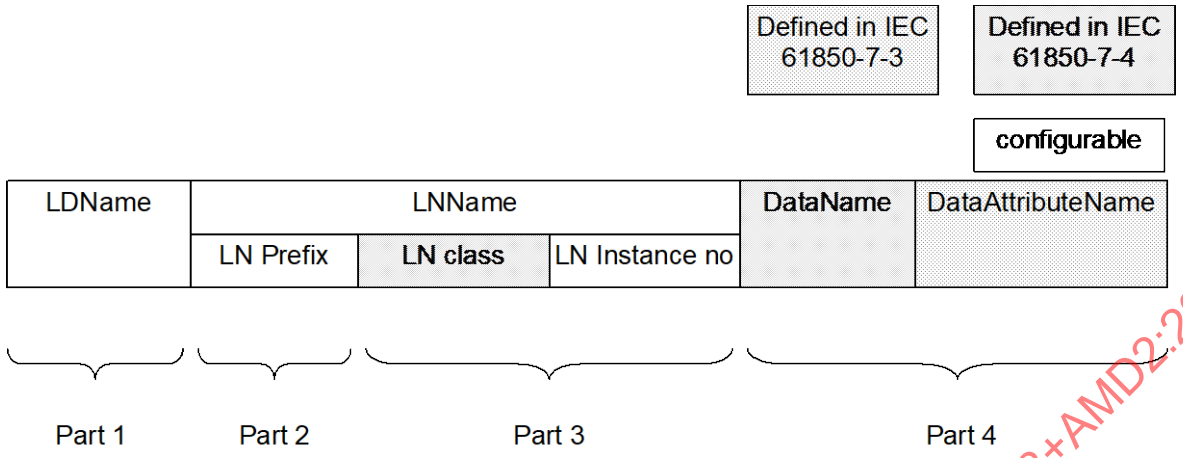


Figure 10 – Elements of the signal identification as defined in IEC 61850-7-2

The name parts 2 and 3 in Figure 10 together form the LN name and distinguish different LN instances within the same LD of an IED. Both are not semantically standardized. A function-related LN Prefix is preferably used during functional engineering, or to bind an instantiated LN on an IED to some process semantics. The LN instance number of the name part 3 shall be used to distinguish instantiated LNs, which are not (already) bound to a process semantic (for example a CSWI which is not bound to some specific switch type, prefix=""), or which have the same non-empty prefix.

The mapping of these signal name parts to actual signal names is stack- and mapping-related and therefore contained in IEC 61850-8-1 and IEC 61850-9-2. From the SCL point of view, it is sufficient to determine the contents of these parts for a specific SA system. However, IEC 61850-8-1 and IEC 61850-9-2 may contain further restrictions on length and contents of name parts.

The *DataTypeTemplates* definition section of the SCL and the standardized names as defined in IEC 61850-7-3 and IEC 61850-7-4 determine the possible values for name parts 3 and 4 in Figure 10. The LN instance number and the prefix are defined in the IED section of the SCL.

For name parts 1 and 2 in Figure 10 there exist several options, the most important two options are illustrated here.

1) **Product-related naming:** As shown in Figure 11, part 1 in Figure 10 is the name of the IED in the IED (product) section, on which the LN is configured, concatenated with the IED relative LD instance identification. Part 2 and 3 are as predefined within the IED.

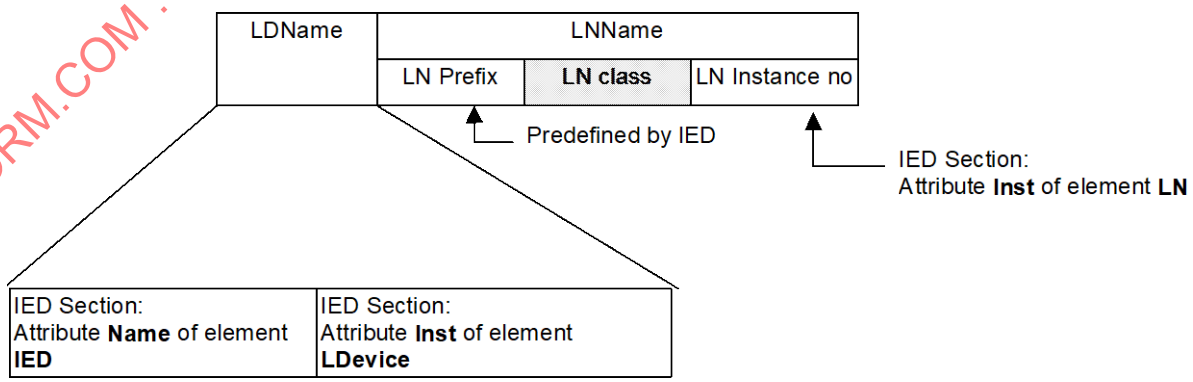


Figure 11 – Elements of the signal name using product naming

2) **Function-related naming:** Function-related naming at communication level is enabled by free setting of the LD name, and possibly free definition of the LN prefix. It is a decision of the IED manufacturer to allow one or both of these options by means of his tools. It has to be kept in mind, that these parts also have to obey special uniqueness restrictions, i.e. can not be used completely free. The following usage could be possible: The LD name, part 1 in Figure 10, is the name of the switch yard function or function type, to which the LN relates. If it is a PrimaryDevice, the name parts from substation name to bay name can be used as part 1, and the PrimaryDevice name (possibly followed by a sub equipment name) can be used in part 2 (LN prefix). If LNs are attached to higher levels than the bay level, naturally the part 1 has to be shortened appropriately, and the part 2 in Figure 10 stays empty, or can be used for the level where the LN is attached to. Observe that according to IEC 61850-7-2, the part 1 (LDName) must be unique within the subnetwork, i.e. it is not allowed to appear on two different IEDs connected to the same subnetwork. So, if you have a main1 protection IED and a main2 protection IED in the same bay E1Q1, each with exactly one logical device for protection, and you use functional naming via the *IdName* attribute, then the LD Name could e.g. be E1Q1F1 for main 1, and E1Q1F2 for main 2 (F stands for protection functions in IEC 81346).

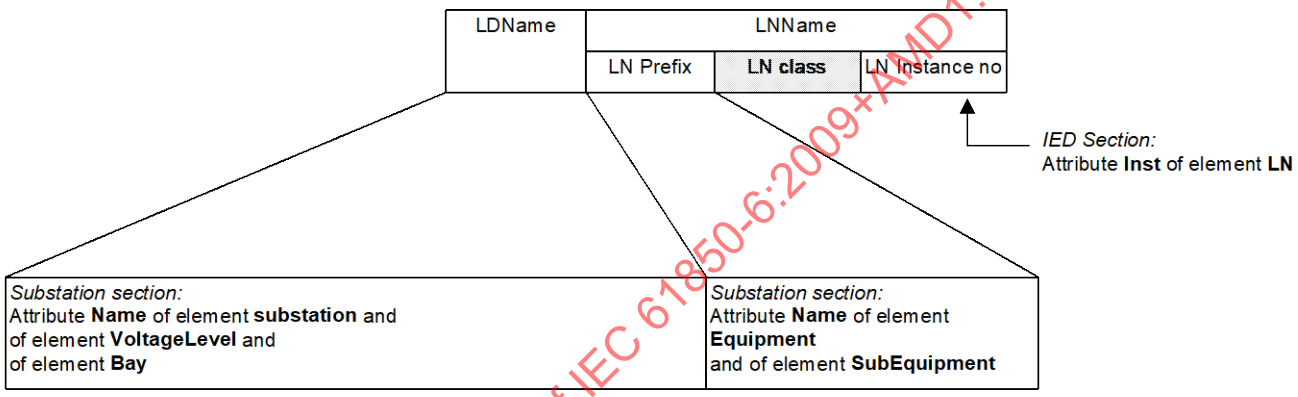


Figure 12 – Possible elements of the signal name using functional naming

The SCL language allows both options, even separate for different IEDs. The mandatory option is the product-related naming. If function oriented naming is needed, the (optional) function oriented LD name has to be explicitly specified for each logical device. It is recommended to use the LN instance number in such a way that the LN class and LN instance number together are always unique. This allows the way of naming (with/without prefix) to be changed at a later time, and even to later replace preconfigured prefixes by prefixes related to the functional structure. The use of these features might be restricted by the IED manufacturer if an IED has a fixed prefix and LN instance number, i.e. does not allow to change this for a certain LN instance later on. In this case function-related naming can be chosen only at LD level. Observe that also the LD inst name part is under control of the IED configurator only and might be fixed for a certain IED type, because it serves as manufacturer identification of the logical device on this IED type. The IED name however and the (function oriented) LD name, if supported at all, shall be freely choosable by a system integrator. In any case, as for product-related naming, the meaning of a LN in the context of the switch yard can be established via the LN link from the substation section to the IEDs.

Observe that SCL internal references to logical nodes and data objects always use the IED-related names, even if another communication-related name (LD name) is defined.

8.5.4 Signal identifications usable by applications

The communication-related names, even if function oriented, depend on manufacturer supplied engineering capabilities as well as the concrete distribution of logical nodes on the IEDs. Applications needing a functional view independent from this should use a signal identification based on the Substation structure names down to the LN class, and then followed by the semantically completely standardized data object and attribute names. A switch position could then be identified by the path name <substation name (AA1)><voltage level name (J1)><bay name (Q1)><Equipment name (QB1)>CSWI.Pos, an earth fault protection function in Main1 e.g. by the path name <substation name (AA1)><voltage level name (J1)><bay name (Q1)><function name (Main1)><Subfunction name (EF1)>PTOC.Op. The SCL language allows this kind of application-related naming in parallel to the communication-related naming, and a complete SCD file might serve as a data base to translate from one to the other.

If several LN instances of the same class shall be allocated to the same equipment, this leads to non unique application related names. To avoid this, additional Function / SubFunction respective EqFunction / EqSubfunction hierarchy levels must be introduced below the common element. In reference to the above example, a main 2 earth fault function PTOC for bay J1Q1 can be allocated to a function Main2 with subfunction EF1.

8.5.5 Naming example

Figure 13 shows an example of an IED with LNs, which control a circuit breaker QA1 of bay Q1 at voltage level E1. The naming is chosen according to the IEC 81346 series. In this example, the IED as a product has the same higher-level product designation part according to the bay (-E1Q1) as the controlled circuit breaker QA1 has in its functional designation (=E1Q1QA1). Figure 13 shows the resulting references within different structures, and the resulting LN reference for communication.

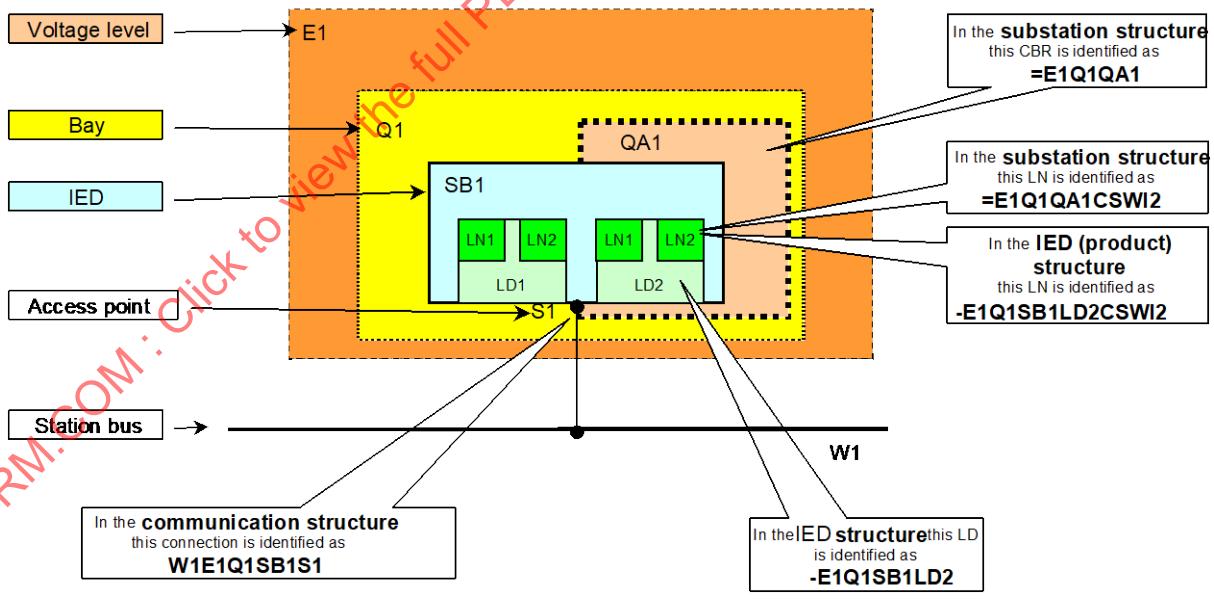


Figure 13 – Names within different structures of the object model

If DATA of LN2 of LN class CSWI within LD2 are now named with names from the function structure, i.e. the LD2 LDName would be the bay name E1Q1, then the LN reference according to IEC 61850-7-2 would be E1Q1/QA1CSWI2. If the references were taken from the product structure, it would be E1Q1SB1LD2/CSWI2. Observe that the whole name in each case shall be unique within the subnetwork, which is the case for both names above. However, in the case of the functional name, the LD reference E1Q1 alone is **not necessarily unique within the subnetwork** (only within the IED). It is the responsibility of the project engineer to assure that there is no other IED with LDName E1Q1 – which restricts the system architecture with functional naming to one IED per bay. The application level functional name E1Q1QA1CSWI however is again unique, and independent from communication level functional or IED-related naming.

8.5.6 Universal Unique Identifier

A UUID shall be generated following specification of ISO/IEC 9834-8. This document will not detail the different algorithms to generate a UUID; this is covered by ISO/IEC 9834-8.

The UUID is optionally available for each SCL element represented by a name or which needs to be referenced to give the possibility to have a fixed identifier independent of the textual name.

If the usage of the UUID is required by a tool, the UUID will be generated by the tool which is creating or instantiating the element for the first time in the context of the usage and shall remain unmodified for the duration of the lifetime of the element.

Different tools may need to generate a UUID for their own needs, but depending on the engineering step, it may be considered as a template UUID or an instance UUID. Different cases may apply, as per the following examples.

By nature, every instance UUID in a SCL shall be unique. This is not the case for template UUID, that could be instantiated several times to reference a UUID defined in an imported file referenced by an SclFileReference element as per definition in 9.1.

An ICT which is creating an element in an ICD (e.g. for an LN) will create a UUID which may be used later by the ICT. On instantiation of the IED in a SCD, the SCT will generate a new instance UUID in the context of the SCD and the existing ICD UUID is to be considered as a template UUID, thus the SCT will have to keep the ICD UUID as a template UUID, to allow comparison with the ICD if needed, by the SCT or the ICT. If the IED is preconfigured by the ICT (an IID is directly created by the ICT instead of an ICD), then the instance UUIDs are directly created by the ICT and preserved by SCT after import.

When an ICT imports an SCD, the UUID present in the SCD shall be preserved. If the ICT adds new elements in the datamodel of the IED, it shall create a UUID for the new elements.

The SST creates a system specification with its own UUID. When the SCT is creating a substation based on an SSD, if a UUID has been generated by the SST assigned to the substation, then the SCT will have to know how to use the original UUID, depending on the use case. If the specification is used as base for the new project (i.e. the specification is already a project specification) the SCT keeps this UUID in the project for the duration of the lifetime of the substation. If the specification is used as a template (i.e. the specification is a piece of project, like a bay, reusable in different projects), the SCT will keep the original UUID from the SSD as a template UUID to allow recognition of the original SSD elements by the SST/SCT.

The rules will be detailed in the relevant parts of the document. Each element which can be referenced will have a UUID and may have a template UUID if its creation was based on the instantiation of a template element identified with a unique identifier. In addition, all possible referring elements will have an optional UUID reference attribute indicating the UUID of the referred element (an instance or a template), when defined. A UUID may also be used within ObjectReference to substitute part of the reference.

When a UUID is not defined, the naming reference shall be used, and when a UUID is defined, the UUID reference as well as the naming reference shall be used and coherency of the file has to be guaranteed by all SCTs.

The UUID attributes are defined as group of attributes defining both *uuid* and *templateUuid* attribute used in all other elements.

Here is the XML schema definition of UUID attributes group:

```
<xs:attributeGroup name="agUuid">
    <xs:attribute name="uuid" type="tUUIDAttribute" use="optional"/>
    <xs:attribute name="templateUuid" type="tUUIDAttribute" use="optional"/>
</xs:attributeGroup>
```

9 The SCL syntax elements

9.1 Header

The header serves to identify an SCL configuration file and its version. The UML diagram given in Figure 14 gives an overview on its structure. All SCL file instances shall have a header element.

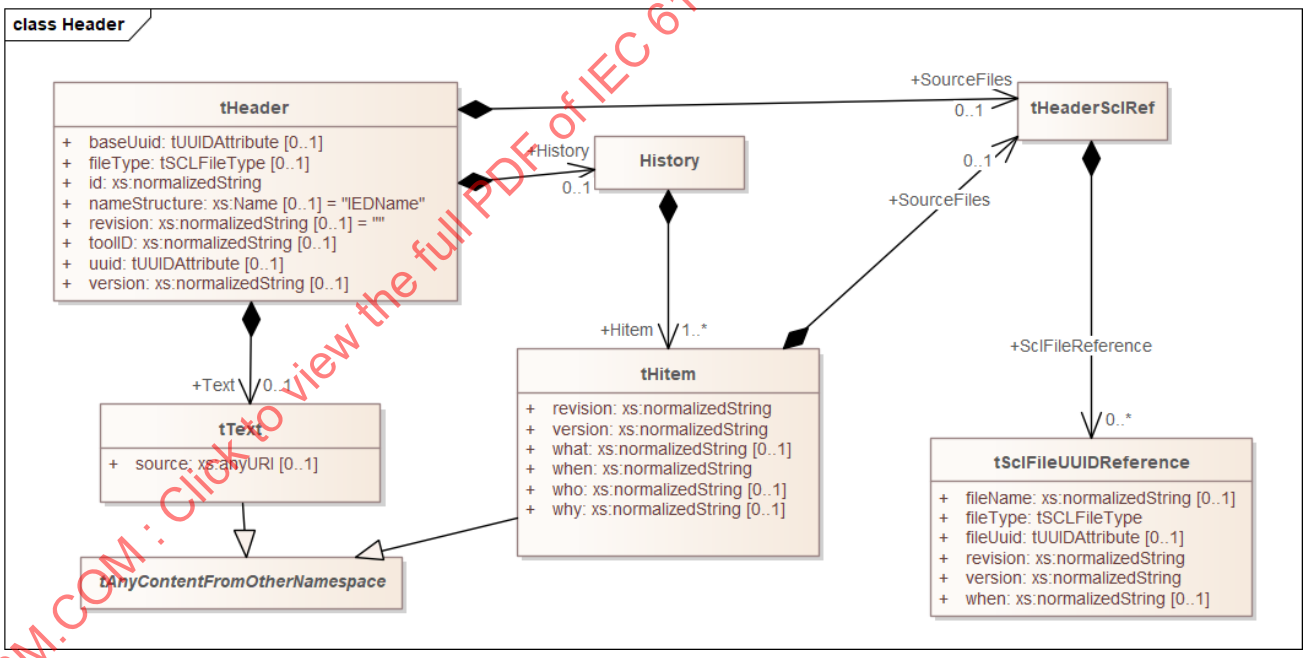


Figure 14 – UML diagram of Header section

Here is the XML schema definition part

```
<xs:complexType name="tHeader">
    <xs:sequence>
        <xs:element name="Text" type="tText" minOccurs="0"/>
        <xs:element name="History" minOccurs="0">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="Hitem" type="tHitem" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="SourceFiles" type="tHeaderSclRef" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:normalizedString" use="required"/>
    <xs:attribute name="version" type="xs:normalizedString"/>
    <xs:attribute name="revision" type="xs:normalizedString" default=""/>
    <xs:attribute name="toolID" type="xs:normalizedString"/>
    <xs:attribute name="fileType" type="tSCLFileType" use="optional"/>
    <xs:attribute name="uuid" type="tUUIDAttribute" use="optional"/>
    <xs:attribute name="baseUuid" type="tUUIDAttribute" use="optional"/>
    <xs:attribute name="nameStructure" use="optional" default="IEDName">
        <xs:simpleType>
            <xs:restriction base="xs:Name">
                <xs:enumeration value="IEDName"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
```

The attributes of the Header element are defined in Table 3.

Table 3 – Attributes of the Header element

Attribute name	Description
id	A string identifying this SCL file, mandatory (can be empty)
version	The project specific version of this SCL configuration file (can be empty, if only one version exists)
revision	The project specific revision of this SCL configuration file, by default the empty string meaning the original before any revision / change.
toolID	The manufacturer specific identification of the tool that was used to create the SCL file
nameStructure	Element provided optional only for backward compatibility with previous SCL schema version. If given at all, only the IEDName value is allowed
fileType	The type of file which has been created (ICD, IID, CID, SSD, SCD, SED or future types). Optional
uuid	A unique identifier generated on creation of the first version of the file or on creation of the project used to identify the file, with different version and revision. Mandatory
baseUuid	The unique identifier of the original file when derived files are generated from this file (i.e. SED files created from a system SCD file).

The file is uniquely identified by a UUID which shall be generated by the tool creating the first version of the file and shall not be altered by any other tool. When a new version of the same file is created, the UUID remains the same, and only the version/revision shall be updated. This will allow the tracking of files during the engineering of systems by identifying the UUID with the version and revision.

A file UUID shall not be shared by different files (different type of file or same type but for different purpose). It is up to the tool to determine if a file is a new version of a previous file or a new file with a new UUID.

When multiple files can be created from the same file, the *baseUuid* attribute is used to identify the original file. This will be used when SED files are created for the same SCD. In this case, the *uuid* attribute will be a new one specific to the SED, and baseUuid will be the UUID of the system file (i.e. the one found in the SCD). This concept could be used in the future for new kind of files.

The element type tSclFileUUIDReference is used, depending on the context where it is declared, and has the following syntax:

```
<xs:complexType name="tSclFileUUIDReference">
    <xs:annotation>
        <xs:documentation>Definition of a reference to an SCL file by its UUID with an associated version revision, corresponding to the definition of an SCL file header</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:attribute name="fileUuid" type="tUUIDAttribute"/>
            <xs:attribute name="fileName" type="xs:normalizedString"/>
            <xs:attribute name="fileType" type="tSCLFileType" use="required"/>
            <xs:attribute name="version" type="xs:normalizedString" use="required"/>
            <xs:attribute name="revision" type="xs:normalizedString" use="required"/>
            <xs:attribute name="when" type="xs:normalizedString"/>
```

```
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes *fileUuid*, *version* and *revision* of a file reference are the one of the Header of the referenced file.

The attributes of the tSclFileUUIDReference element are defined in Table 53.

Table 53 – Attributes of the tSclFileUUIDReference element

Attribute name	Description
fileUuid	The unique identifier attached to the referenced file if defined. Optional.
filename	The name of the file used on import. Optional.
filetype	The type of file which has been used (ICD, IID, CID, SSD, SCD, SED or future types). It depends on the step of the process and is mandatory.
Version	The specific version of the imported SCL file.
Revision	The specific revision of the imported SCL file.
When	The date when the file has been imported. Optional.

Restrictions

- At least *fileUuid* or *fileName* shall be provided in an SCL File reference.

When a file is an SSD or an SCD, the Header element can reference SSD, SCD and/or SED files which has been used to create the SCL file. An SCD can be a reference of another SCD during transfer of a project from one tool to another one. This includes all imported files, even partial SCL files. The newly imported files shall be appended to the end of the list. The history will be represented by the ordering of the element.

For this the Header element has an optional element *SourceFiles* with the following structure to reference the SCL file:

```
<xs:complexType name="tHeaderSclRef">
    <xs:complexContent>
        <xs:extension base="tBaseElement">
            <xs:sequence>
                <xs:element name="SclFileReference" type="tSclFileUUIDReference" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

On creation of a project for specification (SSD) or configuration (SCD) an existing file could be used. Then, if it is decided to continue a previous project, the SourceFiles of the imported file will be kept, otherwise the imported file will be considered as a SourceFile, and the previous files will be cleaned up and the imported file will be identified as a SourceFile itself.

The reference between SCL files mays be defined at Header and/or IED level (see 9.3.2). The different cases of SCL file reference are summarized in Figure 25 and Figure 26.

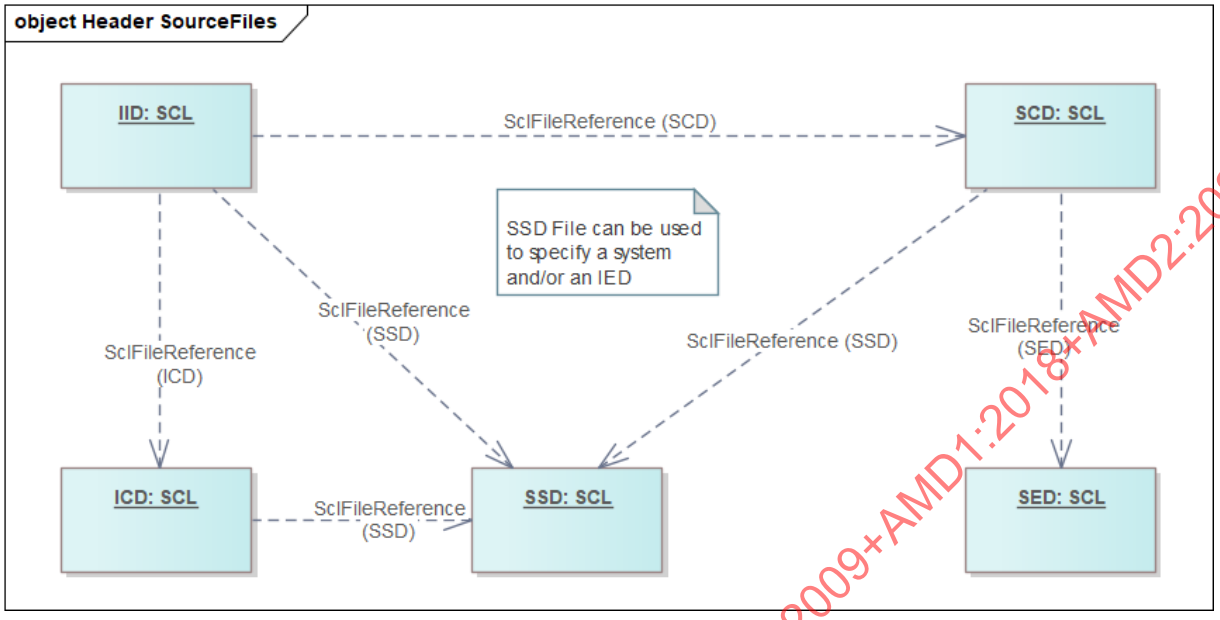


Figure 25 – SCL file references

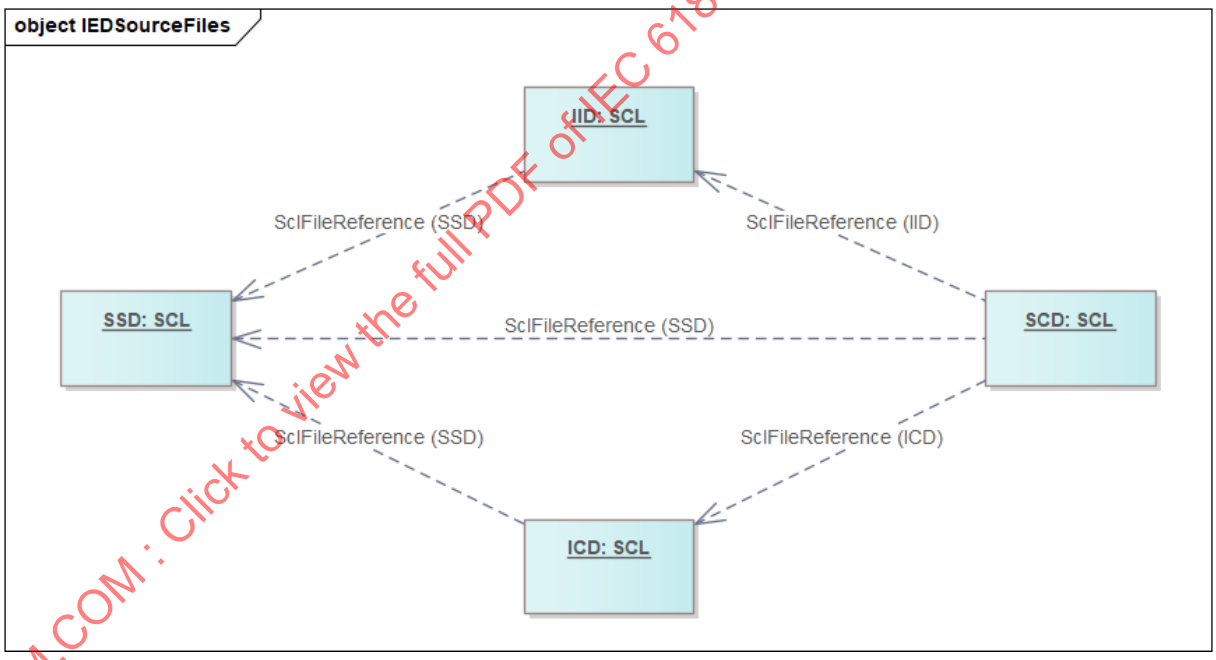


Figure 26 – IED file references

An IID file representing a proxy gateway may contain multiple SCD references, one for each functionality of the gateway, representing each system when the system engineering is done by separate projects. One functionality is the client part of the gateway attached to the substation network (LAN), and another functionality is the server part of the gateway attached to the remote network (WAN).

The *Text* element is optional, and has the following syntax:

```
<xs:complexType name="tText" mixed="true">
    <xs:annotation>
        <xs:documentation xml:lang="en">Allows an unrestricted mixture of character content and element content and attributes from any namespace other than the target namespace.</xs:documentation>
    </xs:annotation>
    <xs:complexContent mixed="true">
        <xs:extension base="tAnyContentFromOtherNamespace">
            <xs:attribute name="source" type="xs:anyURI" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Instead of putting text into this element, a reference to another file can also be given as URI in the *source* attribute.

NOTE The *Text* syntax element for describing text is used in several places, essentially in all elements derived from the *tBaseElement* (see 8.1 and Clause A.1).

The revision history is optional. It describes the modification history of the containing file. This means e.g. that in principle the history of an SCD file is independent from that of IID and ICD files it is based on, although the SCD file history might contain the inclusion of a specific IID file version and revision into a project as comment.

The same syntax can be used also for other documents requiring a revision history. If present, it should have the following form:

```
<xs:complexType name="tHitem" mixed="true">
    <xs:annotation>
        <xs:documentation xml:lang="en"> Allows an unrestricted mixture of character content and element content and attributes from any namespace other than the target namespace, along with the 6 following attributes: Version, Revision, When, Who, What, and Why</xs:documentation>
    </xs:annotation>
    <xs:complexContent mixed="true">
        <xs:extension base="tAnyContentFromOtherNamespace">
            <xs:sequence>
                <xs:element name="SourceFiles" type="tHeaderSclRef" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="version" type="xs:normalizedString" use="required"/>
            <xs:attribute name="revision" type="xs:normalizedString" use="required"/>
            <xs:attribute name="when" type="xs:normalizedString" use="required"/>
            <xs:attribute name="who" type="xs:normalizedString"/>
            <xs:attribute name="what" type="xs:normalizedString"/>
            <xs:attribute name="why" type="xs:normalizedString"/>
```

```
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The history contains several history item entries. Each item identifies a (previously) approved version of this SCL file by means of the attributes described in Table 4. A text within the items can be used to explain further details to this version, and the list of SourceFiles used to create current version may also be indicated, by moving the list of SourceFiles from the header to the Hitem. The global history of the imported files is known by the current list in Header element and the sum of the list in Hitem.

Table 4 – Attributes of the History item (Hitem) element

Attribute name	Description
version	The version of this history entry
revision	The revision of this history entry
when	Date when the version/revision was released
who	Who made/approved this version/revision
what	What has been changed since the last approval
why	Why the change has happened

The following example shows a header without history:

```
<Header id="SCL Example T1-1" toolID="MySystemTool" uuid="6dacf413-cb9e-4ce8-8a45-7c6325b1afaa"
        version="0" revision="1">
    <SourceFiles>
        <SclFileReference fileType="SSD" fileUuid="cef3a885-67d8-44d4-a739-71ebb6f62ad7"
                          fileName="SCLExample.ssd" version="1" revision="0"/>
    </SourceFiles>
</Header>
```

9.2 Process description

9.2.1 General

The process section serves to describe the functional structure of a primary process like a power plant or a substation, and to identify the primary devices and their electrical connections inside substations and power networks. For an industrial process or to describe whole power networks, it is possible to have several substation sections and Line sections modelling the connection between the substations, one substation section for each substation served by the automation system. Sets of substations can be structured by means of the Process element. By means of logical nodes attached to the primary process elements, this clause defines additionally the system functionality (for example, in an SSD file), or, in the case where the logical nodes are already allocated to IEDs (SCD file), the relation of IED functions to the power system.

Note that the *name* attribute is always mandatory and shall not be the empty string. If the substation, line or process section is used as the template within an ICD file, then the name shall be TEMPLATE. The name value at the highest level is also a global identification of the element, because it shall be unique for all elements contained in the SCL file at the same level.

In addition to the *name*, an attribute *uuid* allows the definition of a universal unique identifier (see 8.5.6 for the UUID description). The *uuid* shall be defined by the tool which creates the electrical topology and kept by other tools. As soon as an *uuid* is assigned to a process element, it shall be kept, during the engineering process, either within the attribute *uuid* itself if the process element is an instance within a project, or within the attribute *templateUuid* if the process element is instantiated in a project after the import of the template specification.

If the *desc* attribute is missing, its default value is an empty string.

The *Process* and *Function* related elements have additionally an optional *type* attribute, which allows to define the type of the process / function object. It is recommended that other application areas standardize on these type values.

Logical nodes (LNode) can be attached at each level of the structure (i.e., process, line, substation, voltage level, bay, equipment, subequipment respective function, subfunction). Power transformers (*PowerTransformer*) can also be attached at the structure levels substation, voltage level and bay. Conducting equipments (*ConductingEquipment*) can only be attached to the bay level. Logical node instances at the same level shall have different identifications.

The UML diagram of Figure 15 gives an overview on the substation section:

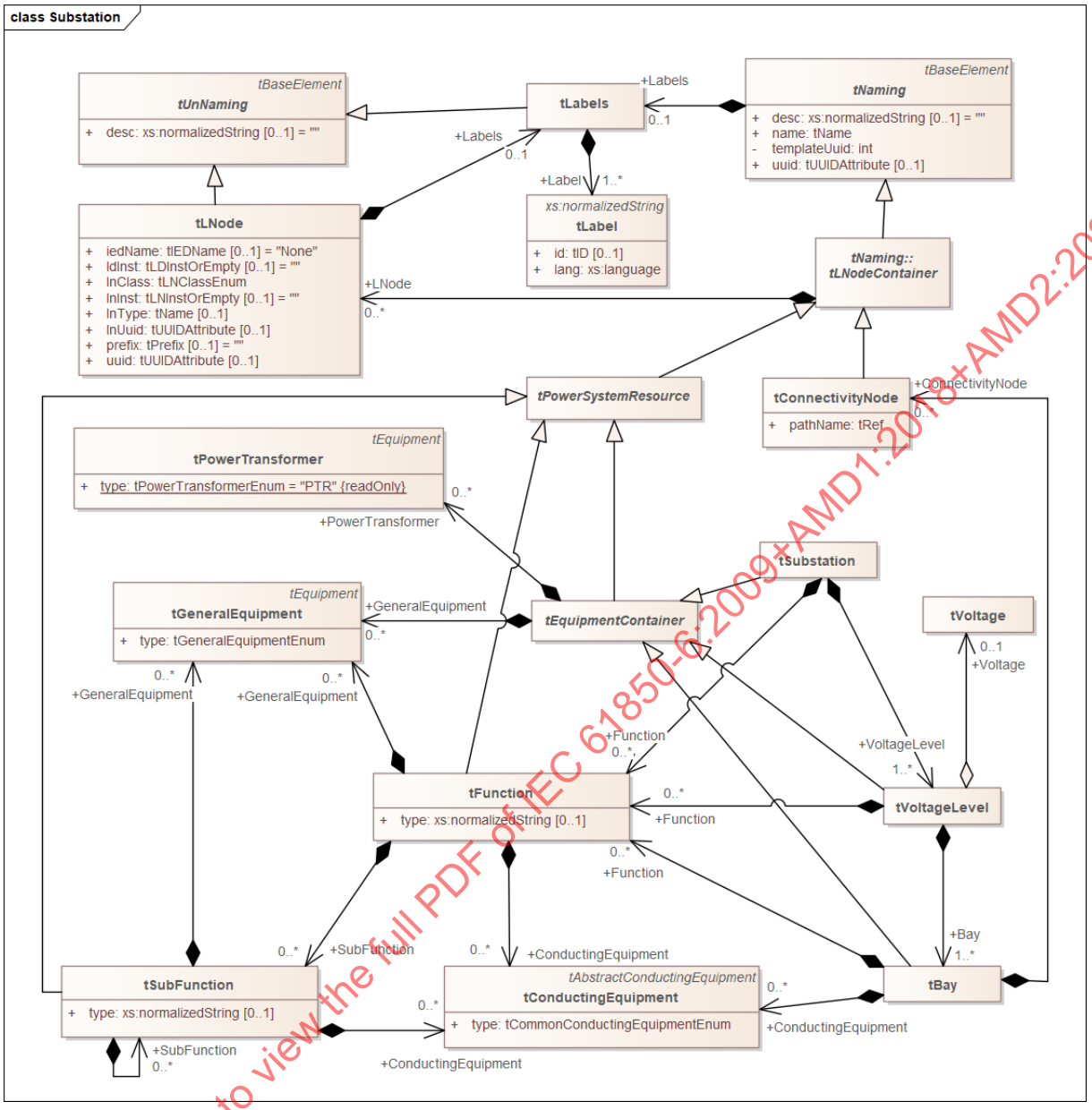


Figure 15 – UML diagram of Substation section

The appropriate schema part is as follows:

These basic type definitions are used for the elements:

```
<xs:include schemaLocation="SCL_BaseTypes.xsd"/>
<xs:attributeGroup name="agVirtual">
    <xs:attribute name="virtual" type="xs:boolean" use="optional" default="false"/>
</xs:attributeGroup>
<xs:complexType name="tLNodeContainer" abstract="true">
    <xs:complexContent>
        <xs:extension base="tNaming">
            <xs:sequence>
```

```
                <xs:element name="LNode" type="tLNode" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tPowerSystemResource" abstract="true">
    <xs:complexContent>
        <xs:extension base="tLNodeContainer"/>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tEquipmentContainer" abstract="true">
    <xs:complexContent>
        <xs:extension base="tPowerSystemResource">
            <xs:sequence>
                <xs:element name="PowerTransformer" type="tPowerTransformer" minOccurs="0" maxOccurs="unbounded">
                    <xs:unique name="uniqueWindingInPowerTransformer">
                        <xs:selector xpath="./scl:TransformerWinding"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="GeneralEquipment" type="tGeneralEquipment" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Then the Substation type is as follows:

```
<xs:complexType name="tSubstation">
    <xs:complexContent>
        <xs:extension base="tEquipmentContainer">
            <xs:sequence>
                <xs:element name="VoltageLevel" type="tVoltageLevel" maxOccurs="unbounded">
                    <xs:unique name="uniqueBayInVoltageLevel">
                        <xs:selector xpath="./scl:Bay"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
```

```
            <xs:unique name="uniquePowerTransformerInVoltageLevel">
                <xs:selector xpath="./scl:PowerTransformer"/>
                <xs:field xpath="@name"/>
            </xs:unique>
            <xs:unique name="uniqueGeneralEquipmentInVoltageLevel">
                <xs:selector xpath="./scl:GeneralEquipment"/>
                <xs:field xpath="@name"/>
            </xs:unique>
            <xs:unique name="uniqueChildNameInVoltageLevel">
                <xs:selector xpath="./*"/>
                <xs:field xpath="@name"/>
            </xs:unique>
        </xs:element>
        <xs:element name="Function" type="tFunction" minOccurs="0" maxOccurs="unbounded">
            <xs:unique name="uniqueSubFunctionInFunctionVL">
                <xs:selector xpath="./scl:SubFunction"/>
                <xs:field xpath="@name"/>
            </xs:unique>
            <xs:unique name="uniqueGeneralEquipmentInFunctionVL">
                <xs:selector xpath="./scl:GeneralEquipment"/>
                <xs:field xpath="@name"/>
            </xs:unique>
        </xs:element>
    </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

The *Substation* element is of type *tSubstation* as shown above. It is an *tEquipmentContainer*, i.e. it might contain logical nodes (LNode) as well as power transformers (*PowerTransformer*). Further it contains at least one voltage level, and optionally several *Function* elements. System functions or equipment, which do not belong to the power system, can be described by the Function element.

The general *Substation* element (of type *tSubstation*), which is referred to by the *SCL* element, includes additionally several identity constraints:

- Within a *Substation*, there cannot be two *VoltageLevel* elements with the same *name*.
- Within a *Substation*, there cannot be two direct *PowerTransformer* elements with the same *name*.
- Within a *Substation*, there cannot be two *Function* elements with the same *name*.

- Within a *Substation*, there cannot be two *LNode* elements with the same combination of *lnInst, lnClass, iedName, ldInst*, and *prefix*.
- Further, in order to avoid any ambiguities, within a *Substation* there cannot be two direct child elements with the same *name*.
- In general, at each hierarchy level within the substation section all names shall be unique, leading to unique object references (path names) of all objects defined by the substation naming hierarchy.

Restrictions

- The substation name shall be unique within an SCL file.
- For a primary system template within an ICD file, the highest level name shall be TEMPLATE. There can be a maximum of one template in one SCL file.
- Within a *Substation* or a *Line*, the attribute *pathName* of a *ConnectivityNode* acts as a key (a *ConnectivityNode* may appear at bay level below the *Substation*, or inside a *Line* element). This implies that there cannot be two *ConnectivityNode* elements with the same *pathName*. The *connectivityNode* attribute of each T*erminal* in this *Substation* or Line must then refer to one of these keys.
- Substation terminals shall not reference connectivity nodes inside Line elements.

The *Process* element is a logical node container, which can be used for other processes than substations, or to group several substations into parts of a power grid. For the first purpose it can also contain equipment elements. It can be recursively used, and its type attribute can indicate the type of the process part / process object identified by the name attribute. It is recommended that the standards for the application areas standardize the type attribute values.

The structure of the Process and Line elements is shown as UML diagram in Figure 24.

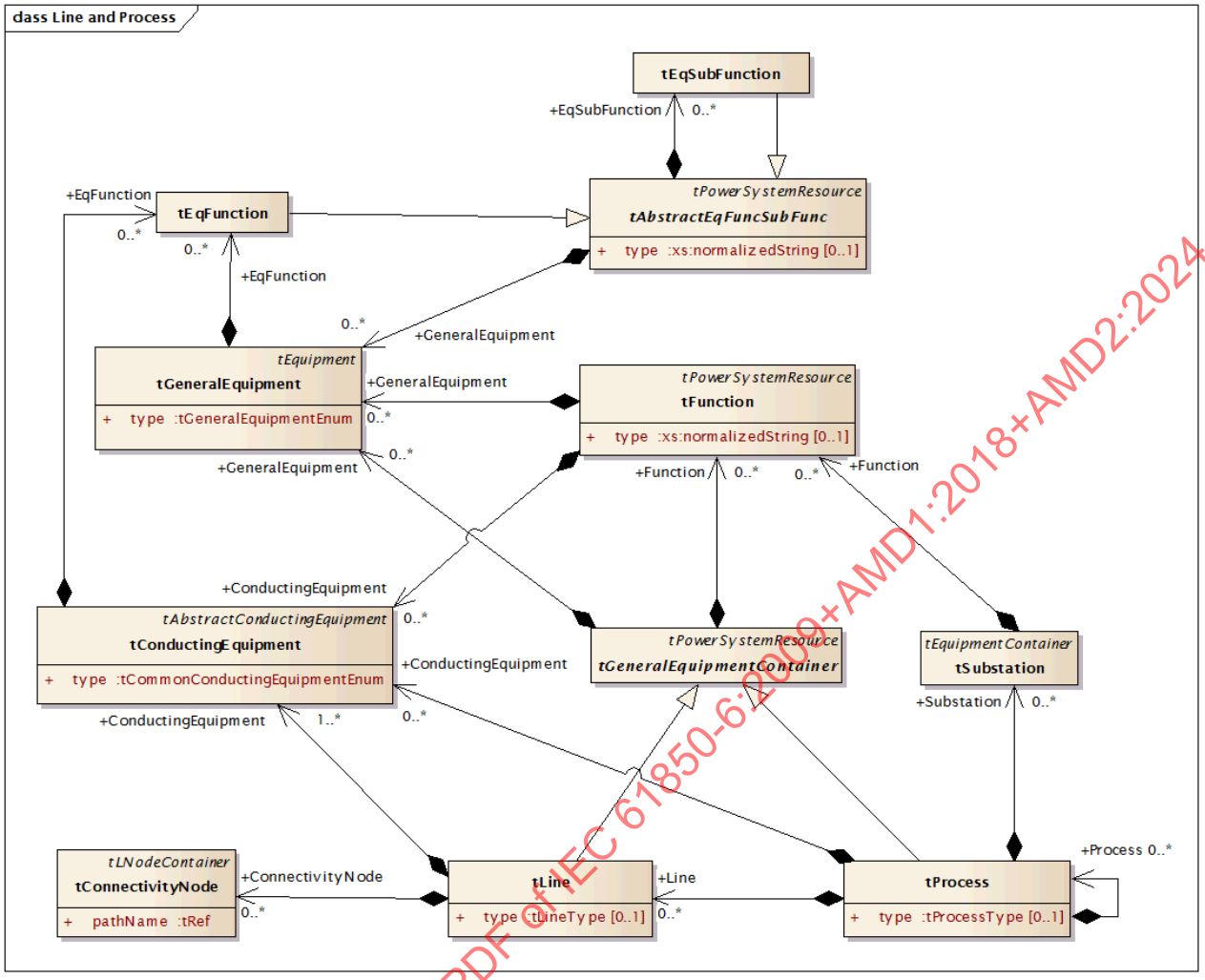


Figure 24 – UML diagram of Process and Line elements

The general *Process* element (of type *tProcess*, see Figure 24), which is referred to by the *SCL* element, includes additionally several identity constraints:

- Within a *Process*, there cannot be two *Substation* elements with the same *name*.
- Within a *Process*, there cannot be two direct *ConductingEquipment* or *GeneralEquipment* elements with the same *name*.
- Within a *Process*, there cannot be two *Function* elements with the same *name*.
- Within a *Process*, there cannot be two *LNode* elements with the same combination of *lnInst, lnClass, iedName, ldInst,* and *prefix.*
- Further, in order to avoid any ambiguities, within a *Process* there cannot be two direct child elements with the same *name*.
- In general, at each hierarchy level within the *Process* section all names shall be unique, leading to unique object references (path names) of all objects defined by the substation naming hierarchy.

Further, inside system specifications, the following rule holds:

- For a primary system template within an ICD file, the Process name shall be TEMPLATE. There can be a maximum of one Process element named TEMPLATE per level in one SCL file.

- Process elements at different levels in a hierarchy branch, which have a type defined, should have different types. Any exceptions shall be explicitly defined for this usage.

The *Line* element is a logical node container, which can be used to model lines between substations of a power grid. It can contain equipment elements modelling line segments, general equipment and connectivity nodes.

The *Line* element has similar restrictions as all other process elements, namely that

- each contained element shall have a unique name.
- if a *Line* element is used as template in an ICD file (i.e. defined as top process element), its name shall be TEMPLATE.

Terminals inside Line elements are allowed to reference connectivity nodes in substations. Substations shall never reference connectivity nodes inside Line elements.

The *Substation*, *Process* and *Line* elements may be identified by a *uuid* and *templateUuid* to be referenced by various SCL elements in the context of a template or instance file.

The *uuid* of an element is independent of its name and shall be kept even if name (or content) of the element is modified. When the electrical topology is modified the elements are considered as new ones and then the *uuid* shall be updated. For example, when a Bay is "moved" from one busbar to another one in the SSD, this is considered as a new Bay and its *uuid* and the *uuids* of all its belonging elements shall be updated to avoid any conflict in tools based on previous version.

9.2.2 Voltage level

A *VoltageLevel* element is of type *tVoltageLevel* as shown below. It has an optional element *Voltage* of type *tVoltage*, which can be used to state the voltage of this voltage level. Furthermore, as *tEquipmentContainer* it might contain logical nodes (*LNode*), *GeneralEquipment* and power transformers (*PowerTransformer*), and it contains one or several bays by means of the *Bay* element, and may contain *Function* elements.

```
<xs:complexType name="tVoltageLevel">
    <xs:complexContent>
        <xs:extension base="tEquipmentContainer">
            <xs:sequence>
                <xs:element name="Voltage" type="tVoltage" minOccurs="0"/>
                <xs:element name="Bay" type="tBay" maxOccurs="unbounded">
                    <xs:unique name="uniqueChildNameInBay">
                        <xs:selector xpath="./*"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                    <xs:unique name="uniqueLNodeInBay">
                        <xs:selector xpath="./scl:LNode"/>
                        <xs:field xpath="@lnInst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@iedName"/>
                        <xs:field xpath="@ldInst"/>
```

```
                <xs:field xpath="@prefix"/>
            </xs:unique>
        </xs:element>
        <xs:element name="Function" type="scl:tFunction" minOccurs="0" maxOccurs="unbounded">
            <xs:unique name="uniqueLNodeInFunctionVL">
                <xs:selector xpath="./scl:LNode"/>
                <xs:field xpath="@lnInst"/>
                <xs:field xpath="@lnClass"/>
                <xs:field xpath="@iedName"/>
                <xs:field xpath="@ldInst"/>
                <xs:field xpath="@prefix"/>
            </xs:unique>
            <xs:unique name="uniqueChildNameInVoltageLevelFunc">
                <xs:selector xpath="./*"/>
                <xs:field xpath="@name"/>
            </xs:unique>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="nomFreq" use="optional">
        <xs:simpleType>
            <xs:restriction base="xs:decimal">
                <xs:minInclusive value="0"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="numPhases" use="optional">
        <xs:simpleType>
            <xs:restriction base="xs:unsignedByte">
                <xs:minExclusive value="0"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Beneath name and desc like any equipment container the voltage level has the following additional attributes:

- *nomFreq*: the nominal frequency in Hz, 0 for DC systems. If missing, the nominal frequency is not known.
- *numPhases*: the number of phases of the single line considered, typically 1, 2 or 3. If missing, not known.

Several identity constraints are defined (in fact, they are defined in *tSubstation* above):

- Within a *VoltageLevel*, there cannot be two *Bay* with the same *name*.
- Within a V*oltageLevel*, there cannot be two direct child *PowerTransformer* elements with the same *name*.
- Within a *VoltageLevel*, there cannot be two direct child *GeneralEquipment* with the same *name*.
- Further, in order to avoid any ambiguities, within a *VoltageLevel*, there cannot be two direct child elements with the same *name*.

Restrictions

- The voltage level name shall be unique within the substation.
- The bay name and function name shall be unique within a voltage level.

The *VoltageLevel* element may be identified by a *uuid* and *templateUuid* to be referenced by various SCL elements in the context of a template or instance file.

9.2.3 Bay level

The *Bay* element is of type *tBay*. As an equipment container, it might contain power transformers, general equipment and logical nodes. Additionally, it might host conducting equipment (*ConductingEquipment*) and connectivity nodes (*ConnectivityNode*), which are used to define topological connections between conducting equipment and power transformers within a single line diagram, and *Function* elements, e.g. for different protection functions.

```
<xs:complexType name="tBay">
    <xs:complexContent>
        <xs:extension base="tEquipmentContainer">
            <xs:sequence>
                <xs:element name="ConductingEquipment" type="tConductingEquipment" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="ConnectivityNode" type="tConnectivityNode" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="Function" type="tFunction" minOccurs="0" maxOccurs="unbounded">
                    <xs:unique name="uniqueSubFunctionInFunction">
                        <xs:selector xpath="./scl:SubFunction"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                    <xs:unique name="uniqueGeneralEquipmentInFunction">
                        <xs:selector xpath="./scl:GeneralEquipment"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
```

```
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The *ConnectivityNode* element allows the explicit definition of connectivity nodes within this bay, and as *tLNodeContainer*, logical nodes (*LNode*) can be attached to it. Its *Text* sub-element can be used to contain some freely usable description. Its *name* attribute identifies the *ConnectivityNode* instance within the bay; its *pathName* is an absolute reference within the SCL file. The pathname is build by all higher level references down to the connectivity nodes name, concatenated with the character "/". For instance, if the connectivity node L1 is within bay Q2 of voltage level E1 of substation Baden, then the pathname is "Baden/E1/Q2/L1".

NOTE 1 The separator "/" has been purposely selected, because the dot "." might appear as part of the names at higher hierarchy levels, for example at bay level.

```
<xs:complexType name="tConnectivityNode">
    <xs:complexContent>
        <xs:extension base="tLNodeContainer">
            <xs:attribute name="pathName" type="tRef" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

NOTE 2 If a bus bar bay does not contain any primary devices, it can be modelled as a bay that contains only connectivity nodes.

Several identity constraints are defined (in fact, they are defined in *tVoltageLevel*):

- Within a *Bay*, there cannot be two direct child elements *PowerTransformer* with the same *name*.
- Within a *Bay*, there cannot be two direct child elements *ConductingEquipment* with the same *name*.
- Within a *Bay*, there cannot be two direct child elements *GeneralEquipment* with the same *name*.
- Further, in order to avoid any ambiguities, within a *Bay*, there cannot be two direct child elements with the same *name*.

The *Bay* element may be identified by a *uuid* to be referenced by various SCL elements in the context of a template or instance file, and the *templateUuid* is used to identify the template used to create the Bay instance if a template has been used. The *ConnectivityNode* element may also be identified by a *uuid* and *templateUuid* to be referenced by a *Terminal*.

An example substation section can be found in 9.2.8.

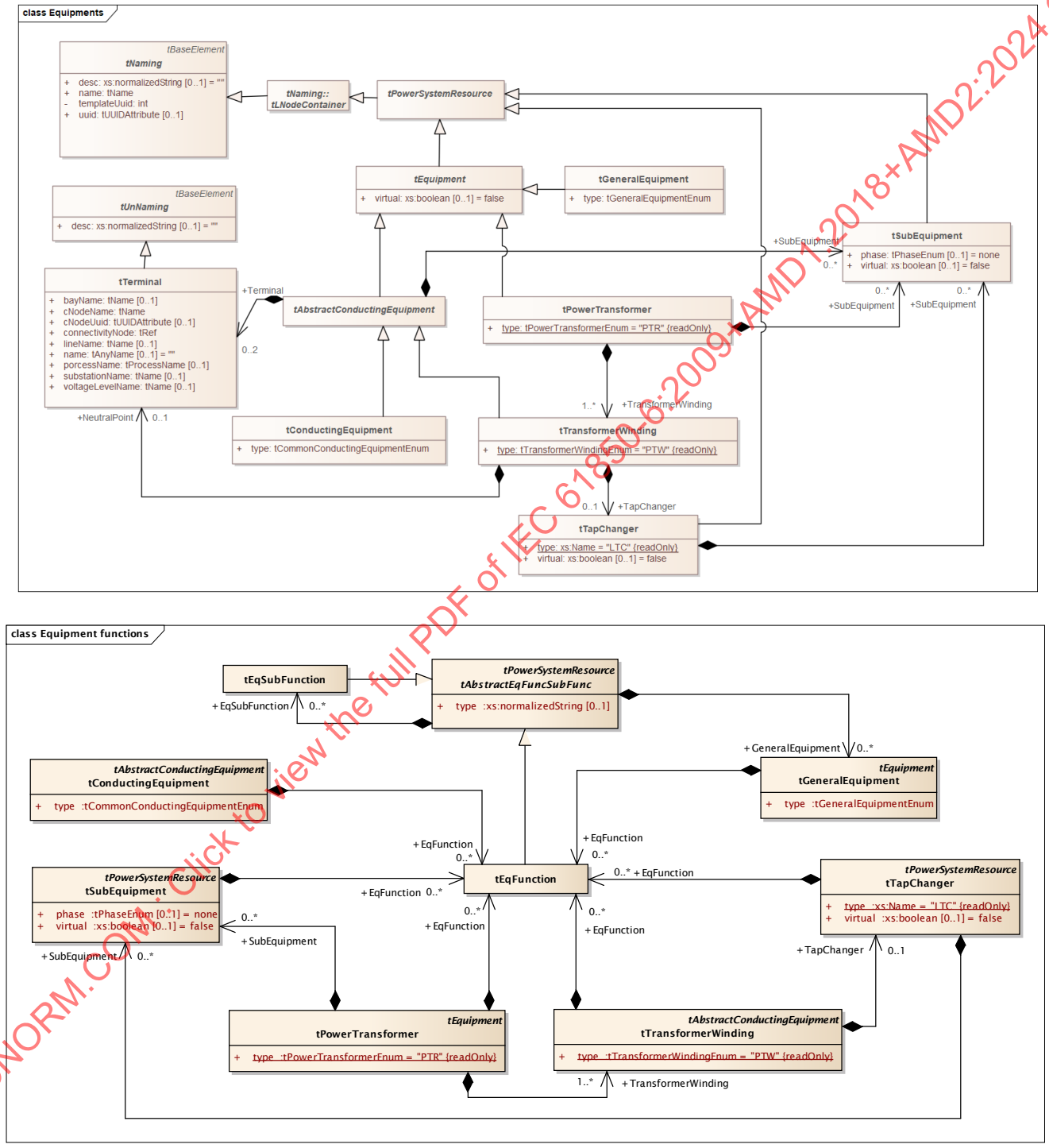
NOTE 3 If no bays are needed within a voltage level, then the whole voltage level can be modelled as just one bay. It has only to be kept in mind, that this virtual bay needs a name of at least one character length.

9.2.4 Power equipment

The power equipment is subdivided into the *PowerTransformer* and *ConductingEqupipment*. The *PowerTransformer* might appear in each equipment container, and contains the transformer windings as special *ConductingEquipment*. To each transformer winding, a tap changer and a

neutral point can be allocated. All other *ConductingEquipment* might appear in the bays only. All equipment is derived from the *tEquipment* base type, and the *ConductingEquipment* from the *tAbstractConductingEquipment* type.

The UML diagram given in Figure 16 gives an overview about the equipment inheritance relations.



IEC

Figure 16 – UML diagrams for equipment type inheritance and relations

The appropriate schema part is as follows.

```
<xs:complexType name="tEquipment" abstract="true">
    <xs:complexContent>
        <xs:extension base="tPowerSystemResource">
            <xs:attributeGroup ref="agVirtual"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tAbstractConductingEquipment" abstract="true">
    <xs:complexContent>
        <xs:extension base="tEquipment">
            <xs:sequence>
                <xs:element name="Terminal" type="tTerminal" minOccurs="0" maxOccurs="2"/>
                <xs:element name="SubEquipment" type="tSubEquipment" minOccurs="0" maxOccurs="unbounded">
                    <xs:unique name="uniqueLNodeInSubEquipment">
                        <xs:selector xpath="./scl:LNode"/>
                        <xs:field xpath="@lnInst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@iedName"/>
                        <xs:field xpath="@ldInst"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                    <xs:unique name="uniqueChildNameInACESubEquipment">
                        <xs:selector xpath="./*"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tConductingEquipment">
    <xs:complexContent>
        <xs:extension base="tAbstractConductingEquipment">
            <xs:sequence>
                <xs:element name="EqFunction" type="scl:tEqFunction" minOccurs="0" maxOccurs="unbounded">
                    <xs:unique name="uniqueLNodeInFuncForCE">
```

```
                        <xs:selector xpath="./scl:LNode"/>
                        <xs:field xpath="@lnInst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@iedName"/>
                        <xs:field xpath="@ldInst"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                    <xs:unique name="uniqueChildNameInFuncForCE">
                        <xs:selector xpath="./*"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="type" type="tCommonConductingEquipmentEnum" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tSubEquipment">
    <xs:complexContent>
        <xs:extension base="tPowerSystemResource">
            <xs:sequence>
                <xs:element name="EqFunction" type="scl:tEqFunction" minOccurs="0" maxOccurs="unbounded">
                    <xs:unique name="uniqueLNodeInFuncForSubEq">
                        <xs:selector xpath="./scl:LNode"/>
                        <xs:field xpath="@lnInst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@iedName"/>
                        <xs:field xpath="@ldInst"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                    <xs:unique name="uniqueChildNameInFuncForSubEq">
                        <xs:selector xpath="./*"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="phase" type="tPhaseEnum" use="optional" default="none"/>
```

```
            <xs:attributeGroup ref="agVirtual"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tPowerTransformer">
    <xs:complexContent>
        <xs:extension base="tEquipment">
            <xs:sequence>
                <xs:element name="TransformerWinding" type="tTransformerWinding" maxOccurs="unbounded">
                    <xs:unique name="uniqueLNodeInTransformerWinding">
                        <xs:selector xpath="./scl:LNode"/>
                        <xs:field xpath="@lnInst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@iedName"/>
                        <xs:field xpath="@ldInst"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                    <xs:unique name="uniqueChildNameInPTW">
                        <xs:selector xpath="./scl:SubEquipment|./scl:TapChanger|./scl:EqFunction"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="SubEquipment" type="scl:tSubEquipment" minOccurs="0" maxOccurs="unbounded">
                    <xs:unique name="uniqueLNodeInSubEquipmentPTR">
                        <xs:selector xpath="./scl:LNode"/>
                        <xs:field xpath="@lnInst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@iedName"/>
                        <xs:field xpath="@ldInst"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                    <xs:unique name="uniqueChildNameInPTRSubEquipment">
                        <xs:selector xpath="./*"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="EqFunction" type="scl:tEqFunction" minOccurs="0" maxOccurs="unbounded">
```

```
                    <xs:unique name="uniqueLNodeInFuncForPTR">
                        <xs:selector xpath="./scl:LNode"/>
                        <xs:field xpath="@lnInst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@iedName"/>
                        <xs:field xpath="@ldInst"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                    <xs:unique name="uniqueChildNameInFuncForPTR">
                        <xs:selector xpath="./*"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="type" type="tPowerTransformerEnum" use="required" fixed="PTR"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tTransformerWinding">
    <xs:complexContent>
        <xs:extension base="tAbstractConductingEquipment">
            <xs:sequence>
                <xs:element name="TapChanger" type="tTapChanger" minOccurs="0">
                    <xs:unique name="uniqueLNodeInTapChanger">
                        <xs:selector xpath="./scl:LNode"/>
                        <xs:field xpath="@lnInst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@iedName"/>
                        <xs:field xpath="@ldInst"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                    <xs:unique name="uniqueChildNameInLTC">
                        <xs:selector xpath="./*"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="NeutralPoint" type="tTerminal" minOccurs="0"/>
```

```
            <xs:element name="EqFunction" type="scl:tEqFunction" minOccurs="0" maxOccurs="unbounded">
                <xs:unique name="uniqueLNodeInFuncForPTW">
                    <xs:selector xpath="./scl:LNode"/>
                    <xs:field xpath="@lnInst"/>
                    <xs:field xpath="@lnClass"/>
                    <xs:field xpath="@iedName"/>
                    <xs:field xpath="@ldInst"/>
                    <xs:field xpath="@prefix"/>
                </xs:unique>
                <xs:unique name="uniqueChildNameInFuncForPTW">
                    <xs:selector xpath="./*"/>
                    <xs:field xpath="@name"/>
                </xs:unique>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="type" type="tTransformerWindingEnum" use="required" fixed="PTW"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="tTapChanger">
  <xs:complexContent>
    <xs:extension base="tPowerSystemResource">
      <xs:sequence>
        <xs:element name="SubEquipment" type="scl:tSubEquipment" minOccurs="0" maxOccurs="unbounded">
          <xs:unique name="uniqueLNodeInSubEquipmentLTC">
            <xs:selector xpath="./scl:LNode"/>
            <xs:field xpath="@lnInst"/>
            <xs:field xpath="@lnClass"/>
            <xs:field xpath="@iedName"/>
            <xs:field xpath="@ldInst"/>
            <xs:field xpath="@prefix"/>
          </xs:unique>
          <xs:unique name="uniqueChildNameInLTCSubEquipment">
            <xs:selector xpath="./*"/>
            <xs:field xpath="@name"/>
          </xs:unique>
        </xs:element>
```

```
            <xs:element name="EqFunction" type="scl:tEqFunction" minOccurs="0" maxOccurs="unbounded">
                <xs:unique name="uniqueLNodeInFuncForLTC">
                    <xs:selector xpath="./scl:LNode"/>
                    <xs:field xpath="@lnInst"/>
                    <xs:field xpath="@lnClass"/>
                    <xs:field xpath="@iedName"/>
                    <xs:field xpath="@ldInst"/>
                    <xs:field xpath="@prefix"/>
                </xs:unique>
                <xs:unique name="uniqueChildNameInFuncForLTC">
                    <xs:selector xpath="./*"/>
                    <xs:field xpath="@name"/>
                </xs:unique>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="type" type="xs:Name" use="required" fixed="LTC"/>
        <xs:attributeGroup ref="agVirtual"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="tGeneralEquipment">
  <xs:complexContent>
    <xs:extension base="tEquipment">
      <xs:sequence>
        <xs:element name="EqFunction" type="scl:tEqFunction" minOccurs="0" maxOccurs="unbounded">
          <xs:unique name="uniqueLNodeInFuncForGE">
            <xs:selector xpath="./scl:LNode"/>
            <xs:field xpath="@lnInst"/>
            <xs:field xpath="@lnClass"/>
            <xs:field xpath="@iedName"/>
            <xs:field xpath="@ldInst"/>
            <xs:field xpath="@prefix"/>
          </xs:unique>
          <xs:unique name="uniqueChildNameInFuncForGE">
          <xs:selector xpath="./*"/>
            <xs:field xpath="@name"/>
          </xs:unique>
```

```
            </xs:element>
          </xs:sequence>
          <xs:attribute name="type" type="tGeneralEquipmentEnum" use="required"/>
        </xs:extension>
      </xs:complexContent>
</xs:complexType>
```

Observe that all equipment of type *tEquipment*, and all subequipment of type *tSubEquipment* as well as the tap changer (*tTapChanger*) also have, beneath the normal *name* and *desc* attributes, an optional *virtual* attribute (*agVirtual*). If the substation section is just used for function-related naming, this is not really used. However, there are some applications where functions (LNs) calculate values belonging to some 'virtual' equipment, for example a phase current is calculated from the measured values of the other two phases. In this case, it is important to know that the third phase CT is only 'virtually' there, and not in reality. This can be indicated by setting the *virtual* attribute to true. Its default value is false.

The equipment elements may be identified by a *uuid* and *templateUuid* to be referenced by various SCL elements in the context of a template or instance file.

To allow functional identification also below equipment level, all equipment elements allow to contain EqFunction definitions. These are essentially the same as the Function / SubFunction structures above equipment, however do not allow another equipment of the same class inside it.

Terminals and their connections to the connectivity nodes (see *tAbstractConductingEquipment*) model the substation topology on the level of a single line, i.e. the number of phases and special connections between phases are not considered here. The maximum number of possible connections to connectivity nodes depends on the terminals available for a device function type. The type codes given in Table 5 for attribute *type* are selected, based as far as possible on IEC 61850-7-4 LN class names.

Table 5 – Primary apparatus device type codes

Type code	Meaning	Number of terminals (connections to different connectivity nodes)
CBR	Circuit Breaker	2
DIS	Disconnector or earthing switch	2
VTR	Voltage Transformer	1
CTR	Current Transformer	2
PTW	Power Transformer Winding	1/2
PTR	Power Transformer	Implicit via windings
LTC	Load Tap Changer	Part of winding
GEN	Generator	1
CAP	Capacitor bank	1/2
REA	Reactor	1/2
CON	Converter	1/2
MOT	Motor	1
FAN	Fan	1
PMP	Pump	1
EFN	Earth Fault Neutralizer (Petersen coil)	1

Type code	Meaning	Number of terminals (connections to different connectivity nodes)
PSH	Power Shunt	2
BAT	Battery	1
BSH	Bushing	2
CAB	Power cable	2
GIL	Gas Insulated Line	2
LIN	Power overhead line or line segment: line segments connected by connectivity nodes form a line. A line segment within a substation could be used to attach for example special LNs, or physical line properties. For a GIS line segment, GIL could be used instead.	2
RES	Neutral resistor	2
RRC	Rotating reactive component	1
SAR	Surge arrester	1
SCR	Semiconductor controlled rectifier	2
SMC	Synchronous Machine	1
TCF	Thyristor controlled frequency converter	2
TCR	Thyristor controlled reactive component	2
IFL	Infeeding line; substation limiting object; models a possibly infeeding power network line outside the substation at the single line border	1

In addition, private types may be used. To allow compatibility with future enhancements of this standard, they shall start with the character E, contain only capital letters, and have at least three letters.

Observe that the second terminal for a power transformer winding is only foreseen for a tapped transformer, while the winding contains a special element for a neutral point connection terminal, to which e.g. (one phase) earthing switches can be connected. Only one neutral point connection terminal is allowed per winding.

A terminal definition contains the reference to a connectivity node to which the equipment is connected (ConnectivityNode in the model of Figure 6), and optionally the name of the equipment terminal, which connects to this connectivity node. As reference to the ConnectivityNode, the path name as well as a list of attributes is used. Both are mandatory. The path name reference allows the verification of the connection consistency already on an XML schema level, while the attribute list is easier to interpret by most tools. To facilitate the update of the link between a terminal and a connectivity node, the attribute cNodeUuid allows the indication of the unique identifier defined to uniquely identify the connectivity node.

```
<xs:complexType name="tTerminal">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:attribute name="name" type="tAnyName" use="optional" default=""/>
            <xs:attribute name="connectivityNode" type="tConnectivityNodeReference" use="required"/>
            <xs:attribute name="processName" type="tProcessName" use="optional"/>
            <xs:attribute name="lineName" type="tName" use="optional"/>
            <xs:attribute name="substationName" type="tName" use="optional"/>
            <xs:attribute name="voltageLevelName" type="tName" use="optional"/>
```

```
            <xs:attribute name="bayName" type="tName" use="optional"/>
            <xs:attribute name="cNodeName" type="tName" use="required"/>
            <xs:attribute name="cNodeUuid" type="tUUIDAttribute" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Table 6 – Attributes of the Terminal element

Attribute name	Description
name	The optional relative name of the terminal at this Equipment. The default is the empty string, which means that the name of the ConnectivityNode is also the terminal identification.
desc	Descriptive text to the terminal
connectivityNode	The pathname of the connectivity node to which this terminal connects. If the Equipment shall not be connected, then the whole Terminal element shall be removed.
processName	The name of a process element containing the connectivityNode (if any – else missing)
lineName	The name of a line element containing the connectivityNode (only used inside Line element)
substationName	The name of the substation containing the connectivityNode (only used to reference a substation, not a line)
voltageLevelName	The name of the voltage level containing the connectivityNode (only used to reference a substation, not a line)
bayName	The name of the bay containing the connectivityNode (only used to reference a substation, not a line)
cNodeName	The (relative) name of the connectivityNode within its container (bay or line)
cNodeUuid	The UUID of the connectivityNode when defined. Optional.

References to connectivity nodes inside a line are only allowed inside Line elements. In this case the lineName attribute is required. For references to connectivity nodes in substations from Substation or Line elements lineName shall not be used, instead substationName, voltageLevelName and bayName are required.

Equipment terminal identifications are in general only needed if the device polarizes the power flow, i.e. the connections are not interchangeable. If the terminal name attribute is left empty, but a terminal designation is needed, then the default is the equipment identification (substationName voltageLevelName bayName equipmentName) together with the connectivity node identification *connectivityNode*.

There is one predefined connectivity node with the name **grounded**. This is used to model earth potential. Thus, an earthing switch is an isolator (equipment type DIS) that is connected on one side to the connectivity node **grounded**. It is up to the generating tool to decide if **grounded** is one single node for the whole substation, or a separate node at each place where connected, or something in between, for example per bay or voltage level, by generating appropriate pathNames.

9.2.5 SubEquipment level

SubEquipment are parts of the power equipment, like a pump is part of a switch, or like a phase of a switch is a part of the whole switch. They especially allow the specification of a phase relation of LNs. Therefore SCL allows SubEquipment only at conducting equipment (elements ConductingEquipment, PowerTransformer, TransformerWinding and TapChanger).

```
<xs:complexType name="tSubEquipment">
    <xs:complexContent>
        <xs:extension base="tPowerSystemResource">
            <xs:attribute name="phase" type="tPhaseEnum" use="optional" default="none"/>
            <xs:attributeGroup ref="agVirtual"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Table 7 – Attributes of the SubEquipment element

Attribute name	Description
name	The identification of the subEquipment relative to the equipment designation (for example L1, if related to phase A)
desc	A textual description of the subEquipment relative to the device
uuid	A unique identifier generated on creation of the SubEquipement instance. Optional.
templateUuid	The template unique identifier of the SubEquipement when created by importing a template file with an identifier already defined. Optional.
phase	The phase to which the subEquipment belongs. The following phase values are allowed: *A, B, C, N* (neutral), *all* (meaning all three phases), *none* (default, meaning not phase-related). The following additional values are only allowed, if the ConductingEquipment above has type VTR: AB, BC, CA, meaning a VT connected in between the appropriate phases.
virtual	Set to *true*, if the subEquipment (for example phase CT) does not exist in reality, but its values are just calculated. Optional, default is *false*

9.2.6 Process function logical nodes

All equipment and equipment containers are also logical node containers. The logical node (abbreviated here as LN) defines the SA function part performed at the appropriate level of the hierarchy. The LNode element identifies the function by specifying a logical node as defined in IEC 61850-5, IEC 61850-7-x or other domain standards. The optional attribute *desc* may contain some operator-related text describing the LN and its usage.

```
<xs:complexType name="tLNode">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInLNode">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
```

```
            </xs:sequence>
            <xs:attribute name="iedName" type="tIEDName" use="optional" default="None"/>
            <xs:attribute name="ldInst" type="tLDInstOrEmpty" use="optional" default=""/>
            <xs:attribute name="prefix" type="tPrefix" use="optional" default=""/>
            <xs:attribute name="lnClass" type="tLNClassEnum" use="required"/>
            <xs:attribute name="lnInst" type="tLNInstOrEmpty" use="optional" default=""/>
            <xs:attribute name="lnType" type="tName" use="optional"/>
            <xs:attributeGroup ref="agUuid"/>
            <xs:attribute name="lnUuid" type="tUUIDAttribute" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The logical node and its function is identified by the element attributes. The *LNode* element can be used within an SSD for functional specification, without allocation to an IED. In this case the *iedName* shall be **None**. This name **None** is forbidden for an IED definition in the IED section. For more detailed specification lnType may refer to a logical node type definition (9.5.2), which then also defines the optional data objects required to exist in this special case, or defines certain values, which some (configuration) parameters shall have. If the logical node is later allocated to an IED within an SCD, then the value of this *lnType* attribute can be ignored, or may be used to check if the logical node type used on the IED fulfills the requirements.

The attributes of the LNode element are described in Table 8.

Table 8 – Attributes of the LNode element

Attribute name	Description
lnInst	The LN instance identification. Can only be missing for lnClass=LLN0, meaning as value here the empty string
lnClass	The LN class as defined in IEC 61850-7-x or other domain standards
iedName	The name of the IED which contains the LN, none if used for specification (default if attribute is not specified)
ldInst	The LD instance on the IED which contains the LN. Within a specification (SSD file), where iedName=None, this shall result in unique LN instance identification, i.e. may contain the LD name
prefix	The LN prefix used in the IED (if needed; default, if not specified, is the empty string). Can be used for more detailed function specification than possible by LN class alone, if the LN is not allocated to an IED
lnType	The logical node type definition containing more detailed functional specification. Might be missing, especially if the LN is allocated to an IED.
uuid	A unique identifier generated on creation of the LNode instance. Optional.
templateUuid	The template unique identifier of the LNode when created by importing a template file with an identifier already defined. Optional.
lnUuid	The UUID of the LN or LN0 which is implementing this LNode, if any. Optional.

NOTE For LLN0, the value of inst is the empty string. In all other cases, it is an unsigned integer.

The *iedName* identifies the IED on which the LN resides, the ldInst the LD within this IED to which the LN belongs. The attributes *prefix*, *lnClass* and *inst* (meaning the LN instance identification according to IEC 61850-7-x or other domain standards) then identify the logical

node within that LD. In this way, the binding between the substation function and the SA system is defined.

Restrictions

- A logical node instance within an IED can only be referenced once within all substation sections.
- Therefore, the combination of *iedName, ldInst, prefix, lnClass* and *lnInst* shall be unique within all substation sections if iedName is not None.
- The naming conventions for all these name parts shall be followed, even if used within a specification.
- Any LN referenced by a LNode shall exist in the file.

For specifications where iedName="None" everywhere, the combination of the other attributes must be unique within the same level. This means e.g. that the prefix or lnInst should be different if several LNs with identical lnClass are used within the same substation part (i.e. same bay). This should also be the case for IED names, if application related naming (see 8.5.4) shall be used. This is NOT checked by the SCL schema, therefore it is the responsibility of the project engineer or system tool, if application related naming shall be used additionally to product (IED)-related naming.

9.2.7 Non power equipment

To be able to model the connection of IED hosted logical nodes to functions other than power system-related ones such as fire fighting equipment or door supervision, the *Process* section as well as the *Substation* sections contain the element *Function*, which again contains an arbitrary number of recursive *SubFunction* elements. Both elements are logical node containers and may also contain *GeneralEquipment*, if necessary. Both *Function* and *Subfunction* have the *name, type* and *desc* attributes, and might also contain the *Text* and *Private* elements. However, there are no connections defined between the equipment. It is possible to have *ConductingEquipment* within these elements to model relations between electrical and non-electrical parts. The *ConductingEquipment* then might contain (electrical) connection definitions to the electrical part.The *SubFunction* element can be used recursively to allow arbitrary function name levels.

```
<xs:complexType name="tFunction">
    <xs:complexContent>
        <xs:extension base="tPowerSystemResource">
            <xs:sequence>
                <xs:element name="SubFunction" type="tSubFunction" minOccurs="0" maxOccurs="unbounded">
                    <xs:unique name="uniqueGeneralEquipmentInSubFunction">
                        <xs:selector xpath="./scl:GeneralEquipment"/>
                        <xs:field xpath="@name"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="GeneralEquipment" type="tGeneralEquipment" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="ConductingEquipment" type="tConductingEquipment" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="type" type="xs:normalizedString" use="optional"/>
        </xs:extension>
```

```
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="tSubFunction">
    <xs:complexContent>
        <xs:extension base="tPowerSystemResource">
            <xs:sequence>
                <xs:element name="GeneralEquipment" type="tGeneralEquipment" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="ConductingEquipment" type="tConductingEquipment" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="SubFunction" type="tSubFunction" minOccurs="0" maxOccurs="unbounded">
            </xs:sequence>
            <xs:attribute name="type" type="xs:normalizedString" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The equipment type allowed within Function and Subfunction is termed GeneralEquipment.

```
<xs:complexType name="tGeneralEquipment">
    <xs:complexContent>
        <xs:extension base="tEquipment">
            <xs:attribute name="type" type="tGeneralEquipmentEnum" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

From the conducting equipment type list (Table 5) this is BAT, MOT, FAN, PMP, additionally to the codes defined in Table 9. Furthermore, private codes (containing only capital letters, starting with "E") can be used. Other parts of this standard or other standards will define more type codes.

Table 9 – General Equipment codes from IEC 61850-7-4

Type code	Meaning
FIL	Filters
VLV	Valves
AXN	Auxillary power network

The *Function*, *SubFunction* and *GeneralEquipement* elements may be identified by a *uuid* and *templateUuid* to be referenced by various SCL elements in the context of a template or instance file.

9.2.8 Substation section example

The following example for a system specification SSD, as shown in Figure 17, contains a substation section for substation Baden220_132 with one transformer T1 between voltage levels D1 and E1, and three bays D1Q1, E1Q2 and E1W1 as bus bar.

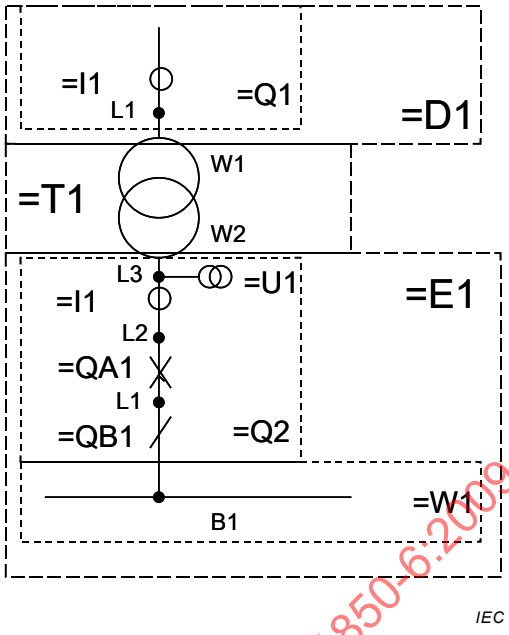


Figure 17 – Substation section example

The transformer T1 has two windings W1 and W2. Winding W1 is connected to a 220 kV voltage level D1 at bay Q1, connectivity node L1. Winding W2 is connected to the bay Q2 in 132 kV voltage level E1. From the attachment of logical nodes in the SSD file it can be seen that there is the measurement of a current transformer at the transformer, and a differential protection. At the 220 kV side (bay D1Q1) there is a distance protection.

The 132 kV bay E1Q2 connects the transformer T1 to the bus bar E1W1. It contains a circuit breaker QA1 and a bus bar disconnector QB1, both electrically connected together at connectivity node L1, as well as a voltage transformer U1 at connectivity node L3, and current transformer I1 between the connectivity nodes L3 and L2. The connectivity node within the same bay is explicitly defined. A logical node of type CSWI controls each switch, and the LN CILO handles the interlocking. No association to IEDs is defined, as this is a functional specification only, so the *iedName* is per default None. In addition, the possibility of defining more details by *lnType* references has not been used here.

NOTE For the example, CT I1 in D1Q1 has only one terminal, which is not allowed in the real system, but accepted for the example.

```
<?xml version="1.0" encoding="UTF-8"?>

  <SCL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.iec.ch/61850/2003/SCL"
  xsi:schemaLocation="http://www.iec.ch/61850/2003/SCL SCL.xsd" version="2007" revision="C" release="5">

      <Header id="SSD Example" uuid="b058aa0f-d8dc-43fd-b953-0e74f53e2b10"/>

      <Substation name="Baden220_132" uuid="0285245f-377f-4ef9-9a87-c4586b3b0aa1">

          <PowerTransformer name="T1" type="PTR" uuid="d3bb4b35-6089-430c-a467-af425560203b">

              <LNode lnInst="1" lnClass="PDIF" ldInst="F1" uuid="23cf9275-b9c9-4029-beff-5c6368ad951f"/>

              <LNode lnInst="1" lnClass="TCTR" ldInst="C1" uuid="0a2b7f4d-a037-4a83-a713-c3bd0c5e9750"/>

              <TransformerWinding name="W1" type="PTW" uuid="1d9a31ae-a7d7-4538-9177-237e7dc040ac">

                  <Terminal connectivityNode="Baden220_132/D1/Q1/L1"
```

```
                substationName="Baden220_132" voltageLevelName="D1" bayName="Q1" cNodeName="L1"
                cNodeUuid="9839cb0b-b4fa-4daf-97f0-5fb7e3676f7d"/>
        </TransformerWinding>
        <TransformerWinding name="W2" type="PTW" uuid="319ad9d1-9589-4854-92d7-c06c4e7752ea">
            <Terminal connectivityNode="Baden220_132/E1/Q2/L3"
                substationName="Baden220_132" voltageLevelName="E1" bayName="Q2" cNodeName="L3"
                cNodeUuid="a5fab2dd-47ca-4d56-8593-e8879f6f3181"/>
        </TransformerWinding>
    </PowerTransformer>
    <VoltageLevel name="D1" nomFreq="50" numPhases="3" uuid="a9439310-f230-4821-8b42-95c32e31eefa">
        <Voltage multiplier="k" unit="V">220</Voltage>
        <Bay name="Q1" uuid="1236774a-ca34-473f-82cc-99cb3c20ec4b">
            <LNode lnInst="1" lnClass="PDIS" ldInst="F1" uuid="d6f2de85-14af-40b5-9065-d916f34a8d46"/>
            <ConductingEquipment name="I1" type="CTR" uuid="a7cd0457-3164-4ded-aedc-e7d974dc81f4">
                <Terminal connectivityNode="Baden220_132/D1/Q1/L1"
                    substationName="Baden220_132" voltageLevelName="D1" bayName="Q1" cNodeName="L1"
                    cNodeUuid="9839cb0b-b4fa-4daf-97f0-5fb7e3676f7d"/>
            </ConductingEquipment>
            <ConnectivityNode name="L1" pathName="Baden220_132/D1/Q1/L1"
                uuid="9839cb0b-b4fa-4daf-97f0-5fb7e3676f7d"/>
        </Bay>
    </VoltageLevel>
    <VoltageLevel name="E1" uuid="8cea2c61-9c6e-4b95-b61b-2f4138844ab6">
        <Voltage multiplier="k" unit="V">132</Voltage>
        <Bay name="Q2" uuid="aa3d78c2-1ff3-4780-b4fc-93262852ec9f">
            <ConductingEquipment name="QA1" type="CBR" uuid="3304b072-8fd3-4047-b7a2-ec2503ad07bb">
                <LNode lnInst="1" lnClass="CILO" ldInst="C1" uuid="3e159ac6-7a94-4754-8402-bf18802c36ab"/>
                <Terminal connectivityNode="Baden220_132/E1/Q2/L1"
                    substationName="Baden220_132" voltageLevelName="E1" bayName="Q2" cNodeName="L1"
                    cNodeUuid="4feaf6f0-34d2-4134-8a17-6707facaf288"/>
                <Terminal connectivityNode="Baden220_132/E1/Q2/L2"
                    substationName="Baden220_132" voltageLevelName="E1" bayName="Q2" cNodeName="L2"
                    cNodeUuid="507ee506-a73d-4f2a-8faf-6e7a0e1f2219"/>
            </ConductingEquipment>
            <ConductingEquipment name="QB1" type="DIS" uuid="27750627-fe77-4914-a7b4-65f2fd4b398b">
                <LNode lnInst="2" lnClass="CSWI" ldInst="C1" uuid="69a616af-2809-4dea-ae3f-3f436f20143f"/>
                <LNode lnInst="2" lnClass="CILO" ldInst="C1" uuid="469ff3a0-3896-41eb-895d-d71dad8bfcb8"/>
```

```
            <Terminal connectivityNode="Baden220_132/E1/W1/B1"
                substationName="Baden220_132" voltageLevelName="E1" bayName="W1" cNodeName="B1"
                cNodeUuid="1fb7decb-5e26-435a-b6e7-113e6a942d3d"/>
            <Terminal connectivityNode="Baden220_132/E1/Q2/L1"
                substationName="Baden220_132" voltageLevelName="E1" bayName="Q2" cNodeName="L1"
                cNodeUuid="4feaf6f0-34d2-4134-8a17-6707facaf288"/>
        </ConductingEquipment>
        <ConductingEquipment name="I1" type="CTR" uuid="458430ca-a64d-4e8b-9f93-6054a69e5deb">
            <Terminal connectivityNode="Baden220_132/E1/Q2/L2"
                substationName="Baden220_132" voltageLevelName="E1" bayName="Q2" cNodeName="L2"
                cNodeUuid="507ee506-a73d-4f2a-8faf-6e7a0e1f2219"/>
            <Terminal connectivityNode="Baden220_132/E1/Q2/L3"
                substationName="Baden220_132" voltageLevelName="E1" bayName="Q2" cNodeName="L3"
                cNodeUuid="a5fab2dd-47ca-4d56-8593-e8879f6f3181"/>
        </ConductingEquipment>
        <ConductingEquipment name="U1" type="VTR" uuid="9dade557-8264-41fb-97a5-3d75a29f55f4">
            <Terminal connectivityNode="Baden220_132/E1/Q2/L3"
                substationName="Baden220_132" voltageLevelName="E1" bayName="Q2" cNodeName="L3"
                cNodeUuid="a5fab2dd-47ca-4d56-8593-e8879f6f3181"/>
        </ConductingEquipment>
        <ConnectivityNode name="L1" pathName="Baden220_132/E1/Q2/L1"
                uuid="4feaf6f0-34d2-4134-8a17-6707facaf288"/>
        <ConnectivityNode name="L2" pathName="Baden220_132/E1/Q2/L2"
                uuid="507ee506-a73d-4f2a-8faf-6e7a0e1f2219"/>
        <ConnectivityNode name="L3" pathName="Baden220_132/E1/Q2/L3"
                uuid="a5fab2dd-47ca-4d56-8593-e8879f6f3181"/>
    </Bay>
    <Bay name="W1">
        <ConnectivityNode name="B1" pathName="Baden220_132/E1/W1/B1"
                uuid="1fb7decb-5e26-435a-b6e7-113e6a942d3d"/>
    </Bay>
  </VoltageLevel>
 </Substation>
</SCL>
```

9.3 IED description

9.3.1 General

The IED section describes the (pre-)configuration of an IED: its access points, the logical devices and the logical nodes instantiated on it. Furthermore, it defines the capabilities of an IED in terms of communication services offered and, together with its LNType, instantiated data (DO) and its default or configuration values. There shall be one IED section for each IED. IED names (name attribute) shall be unique within the file. If only the descriptions of pre-configured IEDs are contained in the file, the name shall be **TEMPLATE** to indicate that the IED has not been bound to a place in the project. The system configurator tool should handle this as an IED type, i.e. a pre–configured product type, from which an arbitrary number of product (hardware) instances can be produced.

NOTE Because the IED *name* is unique within a system, it is also usable as a reference.

A special IED *Router* function is introduced. An IED containing a router function connects different subnetworks by means of all its access points. The router IED may have no logical devices and no logical nodes. In this case, it is managed and supervised by a separate network management system, beyond the scope of this standard. A router is a limiting border, which real time-related message types cannot cross. These message types are:

- time synchronization messages,
- GSE messages,
- sampled analog measurement values.

All other messages are routed through with some time delay.

In addition to the stand-alone router IED described above, the router function can reside on an IED containing additionally clients or servers.

An access point may belong to a server with logical devices, which contain logical nodes. In this case, the server of the access point provides access to the LDs and LNs, while the LNs as clients may use all IED access points (not only those of the server) to access data (on LNs on servers) on other IEDs. An access point always needs a server, if the IED is to be supervised remotely, because the LN0 and LPHD of the server's logical device are used to supervise and control the IED. Only if all LNs on an IED use an access point as a client only, and the IED is not supervised, may an IED without a server be used.

It is recommended that an IED contains at least one server. An access point without a server may then be used to get data from 'lower level' busses, i.e. a bay unit from a process bus. However, this data from the lower level bus cannot be seen directly on the higher level bus unless a router function also resides on this IED. Figure 18 gives a typical example of an IED connected to a station bus and process bus.

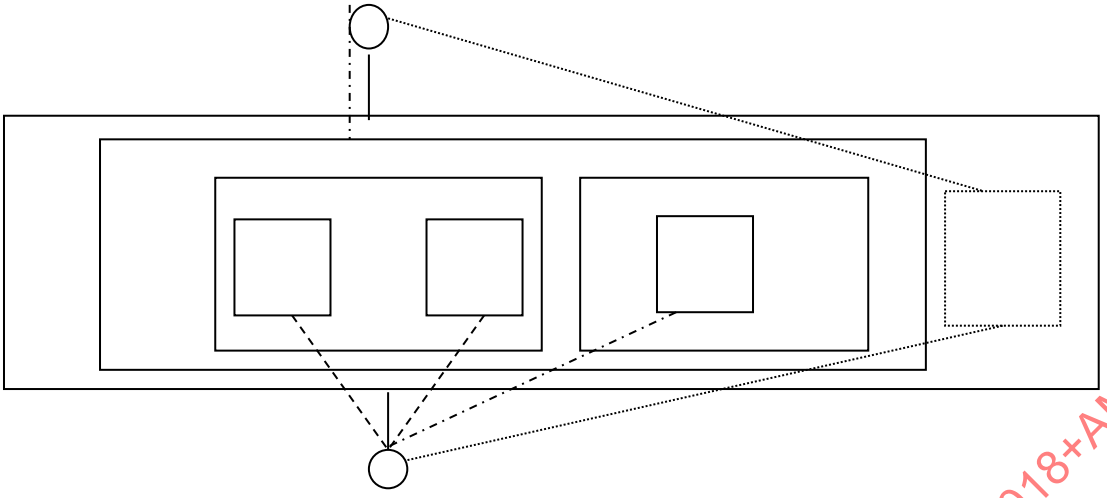


Figure 18 – IED structure and access points

By means of the short address feature, it is possible to define a translation of logical names to short addresses on a data attribute basis.

The usage and meaning of short addresses may be defined in an SCSM (stack mapping). In this case, the system configurator handles them. If an SCSM does not define this, the IED tools might use short address-related attributes as reference to IED internal addresses. In this case, the IED tool handles them. All other tools shall just import and reexport their contents.

Details concerning short addresses can be found in 9.5.4.3.

Figure 19 to Figure 21 give an overview of the IED-related schema part in the form of UML diagrams.

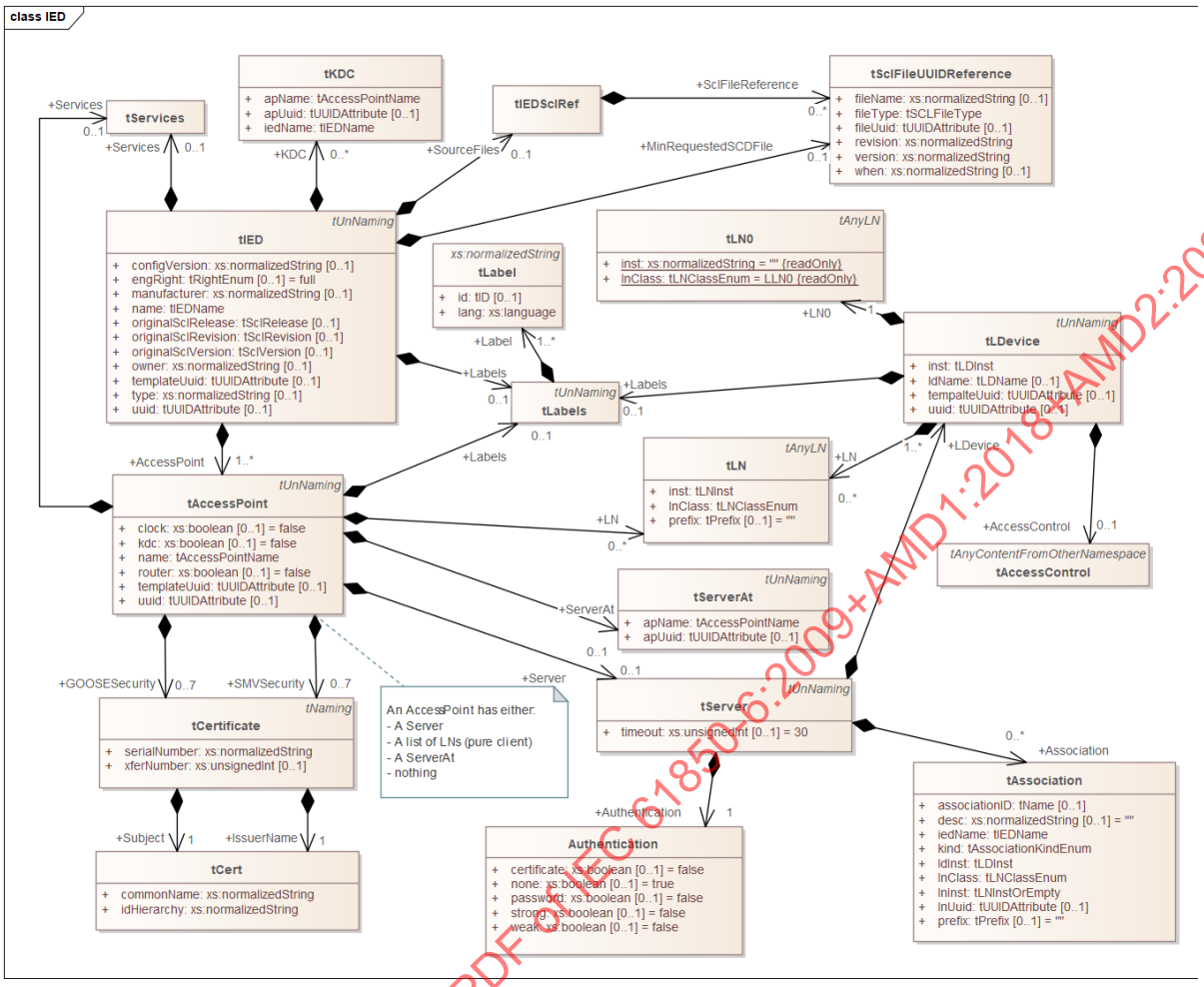


Figure 19 – UML description of IED-related schema part – Base

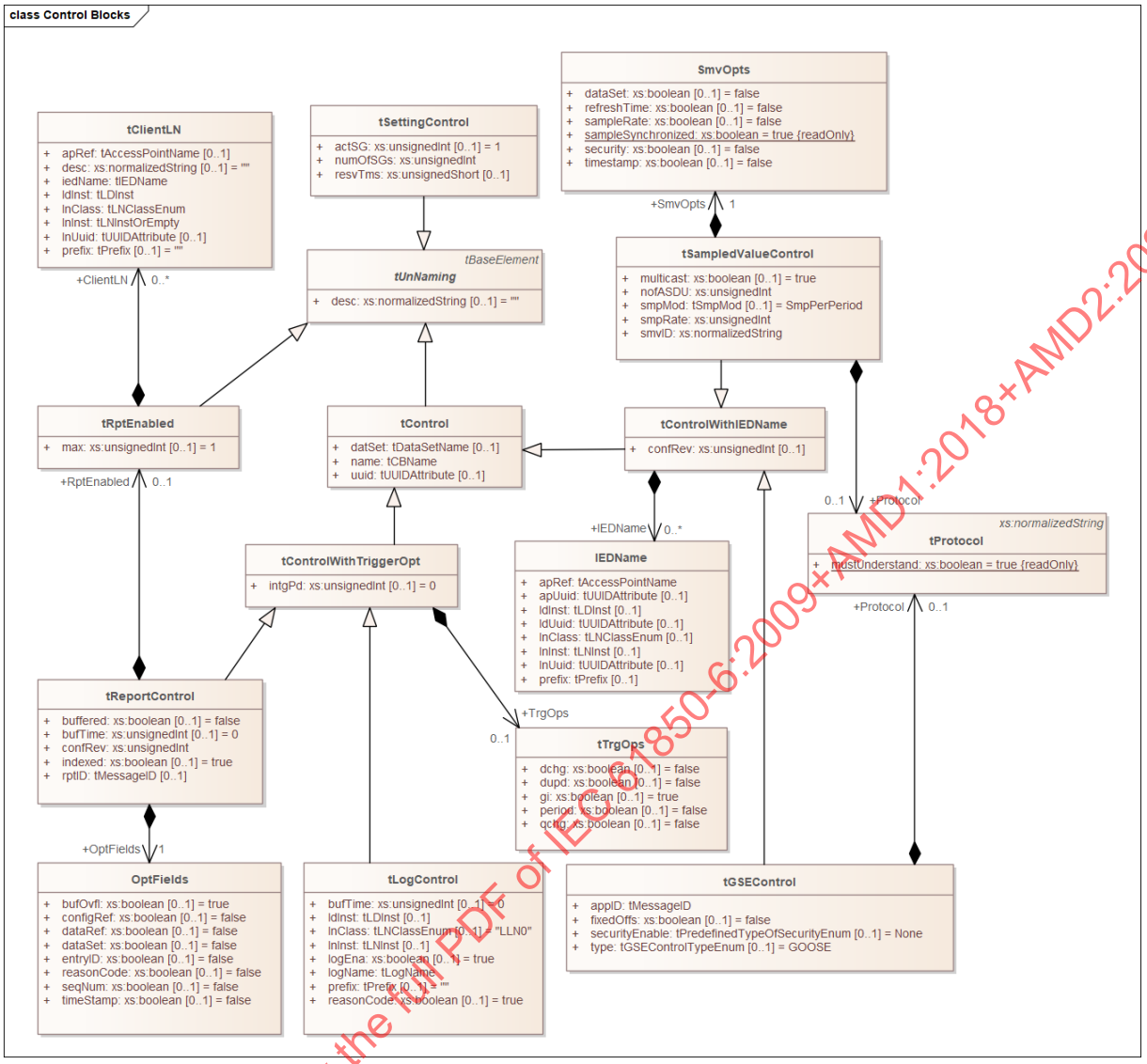


Figure 20 – UML description of IED-related schema part for Control blocks

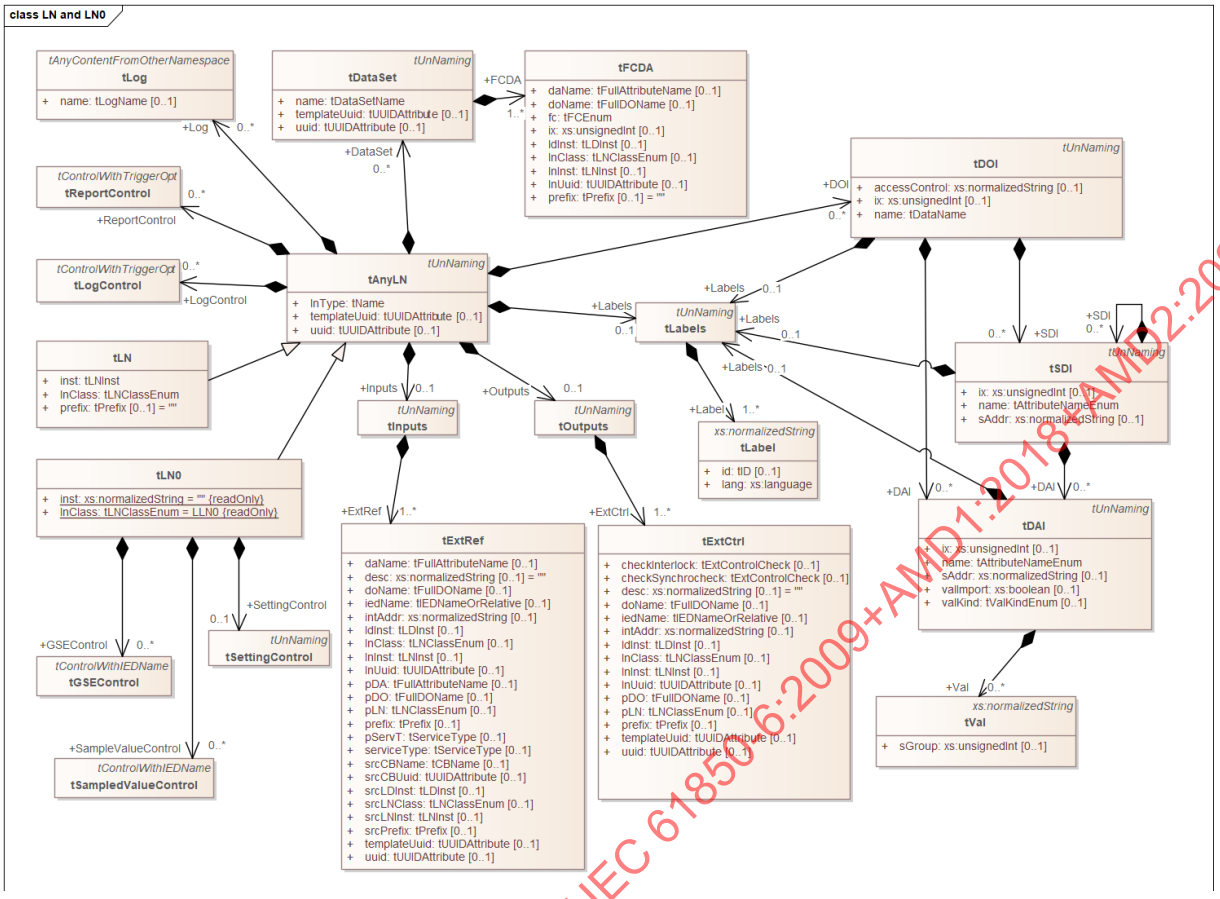


Figure 21 – UML description of IED-related schema part – LN definition

9.3.2 The IED, Services and Access Point

The SCL syntax to describe an IED is as follows:

```
<xs:complexType name="tIED">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Services" type="tServices" minOccurs="0"/>
                <xs:element name="AccessPoint" type="tAccessPoint" maxOccurs="unbounded">
                    <xs:unique name="uniqueLNInAccessPoint">
                        <xs:selector xpath="./scl:LN"/>
                        <xs:field xpath="@inst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="KDC" type="tKDC" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="IEDSourceFiles" type="tIEDSclRef" minOccurs="0"/>
```

```
                <xs:element name="MinRequestedSCDFiles" type="tMinRequestedSCDFiles" minOccurs="0">
                    <xs:unique name="uniqueProjectInReqSCD">
                        <xs:selector xpath="./scl:MinRequestedSCDFile"/>
                        <xs:field xpath="@fileUuid"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInIED">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="tIEDName" use="required"/>
            <xs:attribute name="type" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="manufacturer" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="configVersion" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="originalSclVersion" type="tSclVersion" use="optional" default="2003"/>
            <xs:attribute name="originalSclRevision" type="tSclRevision" use="optional" default="A"/>
            <xs:attribute name="originalSclRelease" type="tSclRelease" use="optional" default="1"/>
            <xs:attribute name="engRight" type="tRightEnum" use="optional" default="full"/>
            <xs:attribute name="owner" type="xs:normalizedString" use="optional"/>
            <xs:attributeGroup ref="agUuid"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tIEDSclRef">
    <xs:complexContent>
        <xs:extension base="tBaseElement">
            <xs:sequence>
                <xs:element name="SclFileReference" type="tSclFileUUIDReference" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tMinRequestedSCDFiles">
```

```
    <xs:complexContent>
        <xs:extension base="tBaseElement">
            <xs:sequence>
                <xs:element name="MinRequestedSCDFile" type="tMinRequestedSCDFile" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tMinRequestedSCDFile">
    <xs:complexContent>
        <xs:restriction base="scl:tSclFileUUIDReference">
            <xs:attribute name="fileType" use="required">
                <xs:simpleType>
                    <xs:restriction base="scl:tSCLFileType">
                        <xs:enumeration value="SCD"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tKDC">
    <xs:attribute name="iedName" type="tIEDName" use="required"/>
    <xs:attribute name="apName" type="tAccessPointName" use="required"/>
    <xs:attribute name="apUuid" type="tUUIDAttribute" use="optional"/>
</xs:complexType>
```

The attributes of the IED element are defined in Table 10.

Table 10 – Attributes of the IED element

Attribute name	Description
name	The identification of the IED. Within an ICD file describing a device type, the name shall be TEMPLATE. The IED name shall not be an empty string and not None and shall be unique within an SCL file
desc	The description text
type	The (manufacturer specific) IED product type
manufacturer	The manufacturer's name
configVersion	The basic configuration version of this IED configuration
originalSclVersion	The original SCL schema version of the IEDs ICD file; optional, default "2003"

Attribute name	Description
originalSclRevision	The original SCL schema revision of the IEDs ICD file; optional, default "A"
originalSclRelease	The original SCL schema release of the IEDs ICD file; optional, default '1': Observe that 2003A had no release at all
engRight	The engineering right transferred by a SED file (only fix, dataflow), or the current state in an SCD file. Values are full, dataflow, fix, the default is full
owner	The owner project of this IED, i.e. the Header id of that SCD file of that project which has the right to use the IED tool for this IED. The default is the Header id of the SCD file containing the IED
uuid	A unique identifier generated on creation of the IED instance. Optional.
templateUuid	The template unique identifier of the IED when created by importing a template file with an identifier already defined. Optional.

The IED configVersion in Table 10 only identifies the IED basic configuration (IED type capabilities as defined/delivered by the manufacturer e.g. after preengineering of a flexibly engineerable IED type), and not its individual configuration after instantiation into a project. The version of the project-specific IED is a parameter of the IED instance, or of its logical nodes. It shall be contained in an SCL file as an attribute value of the attribute *LLN0.NamPlt.configRev*. This means that the value of the *LLN0.NamPlt.configRev* attribute must appear in a Val element of a DAI or DA element.

The *originalSclVersion* states the SCL version of the originally generated and imported ICD file of this IED, *originalSclRevision* the SCL revision and *originalSclRelease* the SCL release of this ICD file. Both attributes are set by the IED tool when creating the ICD or IID file and shall be kept within an SCD file. If they are missing an SCL file of a previous version of the standard, the upgrading rules described in I.4.3.3 shall be applied.

The IED contains a Service capability list, and access point definitions.

The IED may contain a KDC which identifies the key distribution server by its iedName and apName, and optionally the apUuid. The key distribution server concept is defined in IEC 62351-9.

The IED may contain IEDSourceFiles which a list of the different IED files used to engineer the IED in the system. It may use SclFileReference to reference ICD, IID or any file used to create the IED as per definition of an tSclFileUUIDReference defined in 9.1. The ordering of the elements represents the history of the imports, with the latest imported files at the end. The type of the referenced file is indicated in SclFileReference and may also be an SSD, when the IED has been specified by a virtual IED in an SSD file, or a SCD when the IED has been configured from a SCD file.

The IED may also contain MinRequestedSCDFiles which are references to the most recent SCD file for a given project with specific version and revision required to configure an IED. These elements are following tSclFileUUIDReference defined in 9.1 restricted to SCD type of file. When an SCT is updating the system configuration, it will be able to identify the impacts on the IEDs and know which device needs to be updated. Then, on creation of a new SCD file version, the SCT will be able to indicate the version of this new SCD as minimum requirement for the IEDs which need to be updated. Then, the IIDs will keep this information to let SCT verify if the ICT has correctly taken into account the latest SCD.

An IED may need different SCDs to be configured when it is interacting between several projects using SEDs to be configured (this is typically the case of a gateway interfacing between different projects). For this purpose, an IED may have multiple MinRequestedSCDFiles but each of them requires to be different (i.e. the *fileUuid* attribute shall be different).

Restrictions

- The IED name shall be unique within the SCL file. An empty string as well as the reserved name None are forbidden.
- The length of the IED Name shall be at least one, at maximum 64 characters. It starts with an alpha character, and contains only alphanumeric characters and the underscore character. Note that there might be more restrictions in other parts of this standard, in IEDs implemented according to previous versions of this standard, or due to usage of this name at engineering time.
- The IED name for an IED template, i.e. an IED within an ICD file, shall be TEMPLATE.
- When an IED has multiple MinRequestedSCDFiles all of them shall be from different projects, i.e. the *fileUuid* attribute shall be unique in the list

The general *IED* element (of type *tIED*), which is contained within the *SCL* element, additionally includes several identity constraints:

- Within an *IED*, there cannot be two *AccessPoint* elements with the same *name*.
- Within an *IED*, there cannot be two *LDevice* elements with the same *inst*. Moreover, the *inst* attribute of an *LDevice* acts as a key within the *IED* for all references within SCL.

The Services element of the IED defines the available services.

```
<xs:complexType name="tServices">
    <xs:all>
        <xs:element name="DynAssociation" type="scl:tServiceWithOptionalMax" minOccurs="0"/>
        <xs:element name="SettingGroups" type="scl:tSettingGroups" minOccurs="0"/>
        <xs:element name="GetDirectory" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="GetDataObjectDefinition" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="DataObjectDirectory" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="GetDataSetValue" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="SetDataSetValue" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="DataSetDirectory" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="ConfDataSet" type="scl:tServiceForConfDataSet" minOccurs="0"/>
        <xs:element name="DynDataSet" type="scl:tServiceWithMaxAndMaxAttributes" minOccurs="0"/>
        <xs:element name="ReadWrite" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="TimerActivatedControl" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="ConfReportControl" type="scl:tServiceConfReportControl" minOccurs="0"/>
        <xs:element name="GetCBValues" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="ConfLogControl" type="scl:tServiceWithMaxNonZero" minOccurs="0"/>
        <xs:element name="ReportSettings" type="scl:tReportSettings" minOccurs="0"/>
        <xs:element name="LogSettings" type="scl:tLogSettings" minOccurs="0"/>
        <xs:element name="GSESettings" type="scl:tGSESettings" minOccurs="0"/>
        <xs:element name="SMVSettings" type="scl:tSMVSettings" minOccurs="0"/>
        <xs:element name="GSEDir" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="GOOSE" type="scl:tGOOSEcapabilities" minOccurs="0"/>
```

```
        <xs:element name="GSSE" type="scl:tServiceWithMax" minOccurs="0"/>
        <xs:element name="SMVsc" type="scl:tSMVsc" minOccurs="0"/>
        <xs:element name="FileHandling" type="scl:tFileHandling" minOccurs="0"/>
        <xs:element name="ConfLNs" type="scl:tConfLNs" minOccurs="0"/>
        <xs:element name="ClientServices" type="scl:tClientServices" minOccurs="0"/>
        <xs:element name="ConfLdName" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="SupSubscription" type="scl:tSupSubscription" minOccurs="0"/>
        <xs:element name="ConfSigRef" type="scl:tServiceWithMaxNonZero" minOccurs="0"/>
        <xs:element name="ValueHandling" type="scl:tValueHandling" minOccurs="0"/>
        <xs:element name="RedProt" type="scl:tRedProt" minOccurs="0"/>
        <xs:element name="TimeSyncProt" type="scl:tTimeSyncProt" minOccurs="0"/>
        <xs:element name="CommProt" type="scl:tCommProt" minOccurs="0"/>
        <xs:element name="SCSM" type="scl:tSCSM" minOccurs="0"/>
        <xs:element name="Security" type="scl:tSecurity" minOccurs="0"/>
        <xs:element name="MultiAPPerSubNet" type="scl:tServiceYesNo" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="nameLength" use="optional" default="32">
        <xs:simpleType>
            <xs:restriction base="xs:token">
                <xs:pattern value="32"/>
                <xs:pattern value="64"/>
                <xs:pattern value="6[5-9]"/>
                <xs:pattern value="[7-9]\d"/>
                <xs:pattern value="[1-9]\d\d+"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
```

Service classes may appear in arbitrary order. If they do not appear, then the services are not available at the IED. For the meaning of the services, refer to IEC 61850-7-2.

The Services element itself has one attribute stating the supported name length at stack level. The default value is 32 corresponding to the MMS mapping of IEC 61850-8-1 and definitions in IEC 61850-7-2 from 2003. The value for this edition is 64.

The list of service capabilities and setting elements and attributes are described in Table 11.

Table 11 – List of service capabilities and setting elements and attributes

Service capability	Description
ClientServices	Indicates which general service classes this IED can use as a client/subscriber: *goose*, *gsse*, sampled values (*sv*), unbuffered reporting (*unbufReport*), buffered reporting (*bufReport*), reading logs (*readLog*). Default (missing element): supported client services not known (except possibly subscribers from GOOSE/GSSE elements) – look into PICS. Required for 2007 version. A pure client shall set at least one of the options to true. The attribute *supportsLdName* indicates that the client/subscriber understands explicit LD name setting at servers (mandatory for clients/subscribers from Ed2 onwards). Default for an attribute missing: false The following attributes allow to describe further client/subscriber limits. If they are missing at IED and access point level, the limits are not known: *maxAttributes*: the maximal number of data set entries (as sum across all received data sets) supported by the client. *maxReports*: the maximal number of report control blocks the client can subscribe to. *maxGOOSE*: the maximum number of GOOSE messages receivable. *maxSMV*: the maximum number of SV messages receivable. *rGOOSE*: if true, GOOSE subscription at network level (layer 3) is supported; default = false. *rSV*: if true, SV message subscription at network level (layer 3) is supported; default = false. *noIctBinding*: if true, the IED configuration tool cannot bind incoming signals to internal addresses; instead, it provides a template with supported internal addresses for possible incoming data. The binding task must be performed by the system configuration tool. *acceptServerInitiatedAssociation*: if true, the client accepts an association request from the Server; default = false. The contained element *TimeSyncProt* allows to specify the supported time synchronization protocols as a client – details see at TimeSyncProt element below. If it is missing, time synchronization is not supported. The contained GOOSE*McSecurity* and SVMcSecurity elements describe the supported security features at client side for GOOSE or SV subscription as described in IEC 62351-6 – for details see McSecurity service below. When one is missing here, the client supports no multicast related security features. As a GOOSE or SV subscriber it can receive signed messages but ignores the signature. The contained element *Security* allows to specify the supported TPAA security as a client – details see at Security element below. If it is missing, Security is not supported.
DynAssociation	All services for dynamic building of associations. The *max* attribute indicates the guarantied number of dynamic associations which are possible under all circumstances.
SettingGroups: SGEdit ConfSG	Setting group services belong to the setting group control block. If this control block is available, then the setting group service SelectActiveSG for activating a setting group is also available. The capability of online editing (IEC 61850-7-2 services SelectEditSG, ConfirmEditSGValues, SetSGValues) is decided with the SGEdit element. ConfSG definition indicates: - The ability to modify by SCD the values of a SettingGroup values, - The ability to update attribute SettingControl.numOfSGs to reduce the number of used groups, - The configured number must not be higher than the number in the ICD file. Both have the following attribute: resvTms: the value true at SGEdit means that this attribute is online visible at the SGCB; at ConfSG it means that the IED tool accepts configured values from an SCD file. The default value is false.

Service capability	Description
GetDirectory	Services for reading the contents of a server, i.e. the LD and LN directories (all LDs, LNs and DATA of the LNs). This is an option without attributes. Includes the IEC 61850-7-2 services GetServerDirectory, GetLogicalDeviceDirectory, GetLogicalNodeDirectory
GetDataObjectDefinition	Service to retrieve the complete list of all DA definitions of the referenced data that are visible and thus accessible to the requesting client by the referenced LN. It is a service without attributes. Refers to IEC 61850-7-2 service GetDataDefinition
DataObjectDirectory	Service to get the DATA defined in a LN. It is a service without attributes. Refers to IEC 61850-7-2 service GetDataDirectory
GetDataSetValue	Service to retrieve all values of data referenced by the members of the data set. It is a service without attributes. Refers to IEC 61850-7-2 service GetDataSetValues
SetDataSetValue	Service to write all values of data referenced by the members of the data set. It is a service without attributes. Refers to IEC 61850-7-2 service SetDataSetValues
DataSetDirectory	Service to retrieve FCD/FCDA of all members referenced in the data set. It is a service without attributes. Refers to IEC 61850-7-2 service GetDataSetDirectory
ConfDataSet	If ConfDataSet is not specified, then the default value of its *max* attribute is equal to the number of already configured data sets, and they may only be modified. If it is specified, it is possible to configure new data sets up to the defined max, or modify existing ones at configuration time via SCL. The attribute meaning is: max – the maximum number of data sets maxAttributes – the maximum number of FCDA elements (data set entries) allowed in a data set. If an ICD file contains data sets with more elements, the system configurator shall not modify it modify – TRUE means that configured data sets may be modified; default: true Note that the control block settings might also restrict data set modifications even if modify=TRUE.
DynDataSet	Services to dynamically create and delete data sets. Refers to IEC 61850-7-2 services CreateDataSet and DeleteDataSet. The attribute meaning is: max – the maximum number of creatable data sets (including any predefined or preconfigured data sets) maxAttributes – the maximum number of FCDA elements (data set entries) allowed in a dynamically created data set
ReadWrite	Basic data read and write facility; includes the IEC 61850-7-2 services GetData, SetData, and the Operate service, if appropriate data exist. It is a capability without attributes.
TimerActivatedControl	This element specifies that timer activated control services are supported. All other control-related services are specified directly at a DO with the ctlModel attribute. It is a service without attributes.
ConfReportControl	Capability of static (by configuration via SCL) creation of report control blocks. If no value of ReportSettings is fix, the system configurator can delete and add ReportControls up to the defined max value. The attribute's meaning is: max – the maximum number of instantiable report control blocks. If this is equal to the number of configured instances, then no new instances can be created. If it is higher than the number of configured instances, the project engineer is allowed to create more instances for new and existing new types, up to this limit. bufMode – unbuffered, buffered, both; the buffer mode allowed to configure for new control block types. If it is not both and bufConf=false, RCBs of the opposite type shall not be deleted and their mode not be changed. Default: both bufConf – boolean. TRUE means, the *buffered* attribute of report control blocks can be changed via SCL. maxBuf – the maximum number of instantiable buffered control blocks. If it is missing, its value is equal to the max value. If supplied, its value shall be smaller than the max value. All RCBs instantiable by the system configurator are indexed.
GetCBValues	Read values of control blocks. It is a service without attributes

Service capability	Description
ConfLogControl	Capability of static (by configuration via SCL) creation of log control blocks. If no value of LogSettings is fix, the system configurator can delete and add LogControls up to the defined max value. The attribute's meaning is: max – maximum number of instantiable log control blocks
ReportSettings	The report control block attributes for which a change of setting is possible at engineering time or online with services SetURCBValues respective SetBRCBValues: The attribute's meaning is: cbName – control block name (Fix, Conf; default = Fix) datSet – data set reference (Fix, Conf, Dyn, default=Fix; Fix means that both the datSet value and the structure of the referenced data set are fix) rptID – report identifier optFields – optional fields to include in report bufTime – buffer time trgOps – trigger options enable intgPd – integrity period resvTms – if true, the ResvTms attribute exists at all buffered control blocks. In this case, if the BRCB instance is allocated to a client, it should be configured as -1 (reserved), else as 0 (free). owner – if true, the report control block possesses an owner attribute. Default is false
LogSettings	The log control block attributes for which a change of setting is possible at engineering time or with service SetLCBValues: The attribute's meaning is: cbName – control block name (Fix, Conf; default = Fix) datSet – data set reference (Fix, Conf, Dyn, default=Fix; Fix means that both the datSet value and the structure of the referenced data set are fix) logEna – log enable trgOps – trigger options intgPd – integrity period
GSESettings	The GSE control block attributes for which a change of setting is possible at engineering time or with service SetGoCBValues: The attribute's meaning is: cbName – control block name (Fix, Conf; default = Fix) datSet – data set reference reference (Fix, Conf, Dyn, default=Fix; Fix means that both the datSet value and the structure of the referenced data set are fix) appID – application identifier dataLabel – value for the object reference if the corresponding element is being sent (applies only to GSSE control blocks, which are deprecated) *kdaParticipant*: if *true*, the server access point supports KDA (Key Delivery Assurance) as described in IEC 62351-9 for GOOSE, and the McSecurity element for the server shall also be specified. GSESettings allows the following subelement: A contained *McSecurity* element describes the supported security options available at each GOOSE control block as described in IEC 62351-6 – for details see McSecurity service below. If kdaParticipant is true, at least one of this options shall be true.

Service capability	Description
SMVSettings	The SMV control block attributes for which a change of setting is possible at engineering time or with service SetMSVCBValues respective SetUSVCBValues: The attribute's meaning is: cbName – control block name (Fix, Conf; default = Fix) datSet – data set reference (Fix, Conf, Dyn, default=Fix; Fix means that both the datSet value and the structure of the referenced data set are fix) svID – sample value identifier optFields – optional fields to include in sample value message smpRate – sample rate configuration capabilities (Fix, Conf, Dyn, default=Fix) samplesPerSec – samples per second resp. seconds per samples are supported synchrSrcId – inclusion of grand master clock Id (IEC 61850-9-3); default = false nofASDU – the number of ASDUs in the SV message (Fix, Conf, default=Fix) pdcTimeStamp – indicates if the PDC time stamp can be included into the message; default is false kdaParticipant: if *true*, the server access point supports KDA (Key Delivery Assurance) as described in IEC 62351-9, and the McSecurity element for the server shall also be specified. SMVSettings allows the following (sub-)elements: SmpRate – defines the implemented sample rate(s) per period SamplesPerSec – defines the implemented sample rate(s) per second SecPerSamples – defines the implemented seconds between samples If no appropriate elements are defined, the sample rate per period or per second as defined by above attributes is assumed to be freely settable A contained subelement *McSecurity* describes the supported security options available at each SMV control block as described in IEC 62351-6 – for details see McSecurity service below. If kdaParticipant is true, at least one of the options shall be true.
ConfLNs	Describes what can be configured for LNs defined in an ICD file The attribute meanings are: fixPrefix – if false, prefixes can be set/changed; default: false fixLnInst – if false. LN instance numbers can be changed; default: false
ConfLdName	If this element is present, as a server, the IED allows the SCT to define functional LD names (by means of the LDevice ldName attribute)
GSEDir	GSE directory services according to IEC 61850-7-2. This capability has no attributes.
GOOSE	This element shows that the IED can be a GOOSE publisher according to IEC 61850-7-2. If no value of GSESettings is fix, the system configurator can delete and add GoCBs up to the defined max value. The subscriber capability shall be exposed in ClientServices. The attributes meaning is: *max* = maximum number of GOOSE control blocks, which are configurable for publishing. Shall be greater or equal to the number of defined GoCBs. (max=0 at an access point means that this access point does not support GOOSE sending). *fixedOffs* = true indicates that fixed offsets are supported and can be set individually per GoCB. Default: false. *goose*: if true, layer 2 GOOSE sending is supported. Default: true. *rGOOSE*: if true, layer 3 (network level) GOOSE sending is supported. Defautl: false.
GSSE	This element shows that the IED can be a binary data GSSE publisher or subscriber according to IEC 61850-7-2. The attributes meaning is: *max* – maximum number of GSSE control blocks, which are configurable. Max=0 means only GSSE client. GSSE is deprecated since Edition 2.

Service capability	Description
SMVsc	This element shows that the IED can be a Sampled Value publisher according to IEC 61850-7-2. If no value of SMVSettings is fix, the system configurator can delete and add SVCBs up to the defined max value. The subscriber capability shall be exposed in ClientServices. The attributes meaning is: *max* = maximum number of SMV control blocks, which are configurable for publishing. Shall be greater than or equal to the number of defined SVCBs. (max=0 at an access point means that this access point does not support SV sending). *delivery*: indicates supported SV types (multicast, unicast, both); default. multicast. If the value is not both and deliveryConf=false, any existing SVCBs of the opposite type shall not be removed. *deliveryConf*: indicates if above type (multicast, unicast) can be configured by the system configurator; default: false *sv*: if true, layer 2 SV is supported; default: true. *rSV*: if true, layer 3 (network level; multicast only) SV is supported; default false.
FileHandling	Indicates that file handling services are supported. The supported services additionally to all Get… services are defined in the PICS. The following attributes indicate on which protocol the file handling is based. If file handling is supplied at all, at least one of the attributes must be true. mms: MMS based file handling, default=true ftp: FTP based file handling, default=false ftps: FTP with SSL based file handling, default = false
SupSubscription	This element shows the capability to supervise GOOSE or SMV subscriptions. The attribute meaning is: maxGo – maximum number of GOOSE subscription supervision LNs (LN class LGOS) to be instantiated on the IED. Default: only preconfigured LNs can be used. . If >0, at least one instance shall exist in the data model. maxSv – maximum number of SV subscription supervision LNs (LN class LSVS) to be instantiated on the IED. Default: only preconfigured LNs can be used. If >0, at least one instance shall exist in the data model. If the actually instantiated number of any category is less, the system configurator is allowed to add more as needed up to the appropriate max. The created instances shall reference the same LNodeType as an existing instance, have a modified unique LN instance number and be in the same logical device. If this element is missing, only preconfigured supervision LNs are allowed to be used. If maxGo/maxSv are specified >0, the total number of LGOS/LSVS shall not exceed this maximum, in any step of the engineering. The system configurator is then allowed to add and remove LGOS/LSVS respectful to this maximum and keep at least one instance, even when no subscription exists, to allow future engineering of new subscription.
ConfSigRef	This element shows the IED capability to include input references into logical nodes. The attribute meaning is: max – maximum number of input references (e.g. data objects InRef and BlkRef, having CDC ORG) which can be instantiated on the IED. If the actually instantiated number is less and more are needed, they can be created by the ICT up to this limit and a new IID file included by the SCT If this element is missing, only existing preconfigured input reference elements are allowed to be used. ConfSigRef is deprecated from Ed2.2 onwards.
SCSM	Specify which SCSM are supported and specific services. GOOSE and SMV are specified by dedicated services. iec61850_8_1: if true, MMS is supported; default true iec61850_8_2: if true, XMPP is supported; default false serverAssociationInitiation: if true, the server is able to initiate the Association; default false
CommProt	Specifies additional lower layer protocols. Observe that IP V4 always needs to be supported, if IP is used at all. ipv6: if true, IP V6 is supported. Default is false

Service capability	Description
TimeSyncProt	Declares the time synchronization protocols supported by the IED / access point either as time server or as time client if contained inside the ClientServices element, dependent also on the access point role . At least one of them shall be true, if time synchronization is supported at all.: sntp: the SNTP protocol; default is true c37_238: the IEEE 1588 method as defined in C37.238; default is false (deprecated) iec61850_9_3: the IEEE 1588 method as defined in IEC 61850-9-3; default is false other: any other method, e.g. PPS (although no protocol); default is false
RedProt	The types of communication redundancy which are supported on the IED. Default is false (none) hsr: true indicates that HSR is supported by the IED prp: true indicates that PRP is supported by the IED rstp: true indicates that RSTP is supported by the IED.
ValueHandling	Defines the handling allowances of valKind modifications for the system configurator. setToRo: if true, valKind=Set for fc=CF, fc=DC or fc=SP can be modified to RO; default is false.
McSecurity	Defines the supported multicast message security handling options at the server connected to this access point. The actually used option is configured at the control block instances. *signature*: if *true*, calculation of a signature is supported. Default: false. *encryption*: if *true*, message encryption is supported. Default: false.
Security	Define the supported TPAA security handling options at accesspoint level as described in IEC 62351-4. If not defined, Security is not supported. *ACSEAuthentication*: if true, "Association control service element" (ACSE) protocol is supported. Default: false. *E2ESecurity*: if true, End-to-end application security is supported. Default: false.
MultiAPPerSubNet	If this element is present, the IED allows to connect multiple AccessPoints to the same SubNetwork (mandatory only for clients/subscribers from Ed2.2 onwards).
Within an IED capability description, the maximum numbers specified above shall be a guaranteed (minimal) maximum, i.e. this number of elements shall be possible to instantiate respective use under all circumstances, for example even if some dynamic memory allocation allows sometimes to have more elements (than maximum) of one type at the cost of another element type (always at least maximum). The value 0 means that none such element can be used / created.	

There are some setting and configuration capabilities which may be performed online, per SCL configuration, or just have fix values. These are indicated by the appropriate attribute values of *Dyn* (dynamically settable by IEC 61850 communication services), *Conf* (configurable via an SCL file), and *Fix* (only one fix value, typically documented in the SCL file). The *Dyn* option in this case always includes the *Conf* option, i.e. if online setting is possible, also setting via SCL shall be possible. If *Fix* applies to a data set reference, also the structure and definition of the referenced data set is fix. For consistency in this case the appropriate ConfReportControl, ConfLogControl, GOOSE and SMVsc elements shall not be supplied. If they are supplied, their max attribute shall be ignored for compatibility with earlier versions of the standard, and no control block instance of the appropriate class shall be removed or added. Additionally any IED configurator supplied control block having values which are NOT allowed to be configured by the system configurator according to the settings, shall keep these values and shall not be removed.

The Access point element of the IED defines the available communication access points.

```
<xs:complexType name="tAccessPoint">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:choice minOccurs="0">
                    <xs:element name="Server" type="scl:tServer">
                        <xs:unique name="uniqueAssociationInServer">
                            <xs:selector xpath="./scl:Association"/>
                            <xs:field xpath="@associationID"/>
                        </xs:unique>
                    </xs:element>
                    <xs:element ref="scl:LN" maxOccurs="unbounded"/>
                    <xs:element name="ServerAt" type="tServerAt"/>
                </xs:choice>
                <xs:element name="Services" type="scl:tServices" minOccurs="0"/>
                <xs:element name="GOOSESecurity" type="tCertificate" minOccurs="0" maxOccurs="7"/>
                <xs:element name="SMVSecurity" type="tCertificate" minOccurs="0" maxOccurs="7"/>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInIED">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="tAccessPointName" use="required"/>
            <xs:attribute name="router" type="xs:boolean" use="optional" default="false"/>
            <xs:attribute name="clock" type="xs:boolean" use="optional" default="false"/>
            <xs:attribute name="kdc" type="xs:boolean" use="optional" default="false"/>
            <xs:attributeGroup ref="agUuid"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The Access point is decribed by one of the elements: *Server, ServerAt* or *LN* list. It may optionally contain the security-related elements GOOSESecurity and SMVSecurity, stating the certificate information to be used for GOOSE sending and Sampled Value sending. The detailed meaning of these certificate descriptions can be found in IEC 62351-6.

The attributes of the Access point element are defined in Table 12

Table 12 – Attributes of the Access point element

Attribute name	Description
name	Reference identifying this access point within the IED
desc	The description text
router	The presence and setting to true defines this IED to have a router function. By default, its value is false (no router function).
clock	The presence and setting to true defines this IED to be a master clock at this bus. By default, its value is false (no master clock).
uuid	A unique identifier for the access point generated on creation of the instance (instance of the IED). Optional.
templateUuid	The template unique identifier of the access point when created by importing a template file with an identifier already defined. Optional.

The name attribute of the access point together with the name of the IED gives a unique reference for the access point within the SA system.

If neither a router, nor a clock, nor a server, nor a LN list is specified, the access point may only be used by client LNs in the same IED to access the bus to which it is connected. This is typical for a process bus access point of a bay level device, where the LNs offer their data via a server to the station bus only.

The Services element specifies the service capabilities of this access point additionally to those already stated at the IED level. It shall not contain IED general configuration or engineering-related capabilities like ConfLNs, ConfLdName or ValueHandling – these shall be defined at IED level. As general rule each element of the Services section shall only be used either at IED level, being common for all servers and access points, or at access point level, being then valid only for this access point. If it is not defined at any level, the defaults as specified above have to be taken. Observe that a Server access point and a ServerAt element at the same server shall not define conflicting service capabilities to the server, i.e. they might sometimes have to repeat the same options. If different access points define different numbers (e.g. number of control blocks accessible via the access point), then for the server the sum is valid. The following table makes this rule visible for each service element and attribute. The column IED marks those elements which are always valid for the whole IED. The column Server versus SeverAt defines the consequences if several access points to the same server exist. The basic idea is that they might have different service classes (client server, GOOSE, SV), however if they have the same server class, the capabilities must be identical.

It should be noted that the meaning of the max attribute of the ConfReportControl, GOOSE and SMVsc elements at an access point. especially for multiple access points at the same server, changes its semantics from 'configurable' to 'capable of sending'. To make it simple, the 'capability of sending' can either be 0 (i.e. no sending) or all control blocks defined. Therefore these elements, if defined at IED level to indicate configuration capabilities, are additionally allowed at AP level, but only with value max=0.

Table 50 shows which service capabilities can be defined at which level. The IED column declares if only at IED (y) or also at access point level (n), and the Server / ServerAt column, which one can be different to the referred server at a ServerAt element.

Table 50 – Usage of Service element at IED level and Server / ServerAt level

Services			IED	Server + ServerAt
nameLength			y	
DynAssociation	max		n	may be different
SettingGroup	SGEdit		n	identical
	SGEdit	resvTms	n	identical
	ConfSG		n	identical
	ConfSG	resvTms	n	identical
GetDirectory			n	identical
GetDataObjectDefinition			n	identical
DataObjectDirectory			n	identical
GetDataSetValue			n	identical
SetDataSetValue			n	identical
DataSetDirectory			n	identical
ConfDataSet	max		n	identical
	maxAttributes		n	identical
	modify		n	identical
DynDataSet	max		n	may be different
	maxAttributes		n	may be different
ReadWrite			n	may be different
TimerActivatedControl			n	identical
ConfReportControl	max		n	identical or 0
	bufMode		n	identical
	bufConf		n	identical
	maxBuf		n	identical
GetCBValues			n	identical
ConfLogControl	max		n	identical
ReportSettings	cbName		n	identical
	datSet		n	identical
	rptID		n	identical
	optFields		n	identical
	bufTime		n	identical
	trgOps		n	identical
	intgPd		n	identical
	resvTms		n	identical
	owner		n	identical
LogSettings	cbName		n	identical
	datSet		n	identical
	logEna		n	identical
	trgOps		n	identical
	intgPd		n	identical

Services			IED	Server + ServerAt
GSESettings	cbName		n	identical
	datSet		n	identical
	appID		n	identical
	dataLabel		n	identical
	kdaParticipant		n	may be different
	McSecurity	signature	n	may be different
	McSecurity	encryption	n	may be different
SMVSettings	cbName		n	identical
	datSet		n	identical
	svID		n	identical
	optFields		n	identical
	smpRate		n	identical
	samplesPerSec		n	identical
	pdcTimeStamp		n	identical
	synchSrcId		n	identical
	nofASDU		n	identical
	SmpRate		n	may be different
	SamplesPerSec		n	may be different
	SecPerSamples		n	may be different
	kdaParticipant		n	may be different
	McSecurity	signature	n	may be different
	McSecurity	encryption	n	may be different
GSEDir			n	identical
GOOSE	max		n	identical or 0
	fixedOffs		n	may be different
	goose		n	may be different
	rGoose		n	may be different
SMVsc	max		n	identical or 0
	delivery		n	may be different
	deliveryConf		n	may be different
	sv		n	may be different
	rSV		n	may be different
FileHandling	mms		n	may be different
	ftp		n	may be different
	ftps		n	may be different
ConfLNs	fixPrefix		y	
	fixLnInst		y	

Services			IED	Server + ServerAt
ClientServices	goose		n	may be different
	gsse		n	may be different
	bufReport		n	may be different
	unbufReport		n	may be different
	readLog		n	may be different
	sv		n	may be different
	rGOOSE		n	may be different
	rSV		n	may be different
	noIctBinding		y	
	acceptServerInitiatedAssociation		n	may be different
	supportsLdName		n	may be different
	maxAttributes		n	may be different
	maxReports		n	may be different
	maxGOOSE		n	may be different
	maxSMV		n	may be different
	TimeSyncProt	sntp	n	may be different
	TimeSyncProt	iec61850_9_3	n	may be different
	TimeSyncProt	other	n	may be different
	GOOSEMcSecurity	signature	n	may be different
	GOOSEMcSecurity	signature	n	may be different
	SVMcSecurity	signature	n	may be different
	SVMcSecurity	encryption	n	may be different
	Security	ACSEAuthentication	n	may be different
	Security	E2ESecurity	n	may be different
ConfLdName			y	
SupSubscription	maxGo		n	identical
SupSubscription	maxSv		n	identical
ConfSigRef	max		n	identical
ValueHandling	setToRO		y	
RedProt	hsr		n	may be different
	prp		n	may be different
	rstp		n	may be different
SCSM	Iec61850_8_1		n	may be different
	Iec61850_8_2		n	may be different
	serverAssociationInitiation		n	may be different
TimeSyncProt	sntp		n	may be different
	iec61850_9_3		n	may be different
	other		n	may be different
CommProt	ipv6		n	may be different
Security	ACSEAuthentication		n	may be different
	E2ESecurity		n	may be different
MultiAPPerSubNet			y	

Project-specific access point attributes, such as the address within a communication system, are contained in the SCL Communication section.

The *ServerAt* element references an existing access point, which shall contain a server. It can be used to define another access point to the same server. It has to be borne in mind that all access points share all control block instances of the defined server. This means, especially, that if a GOOSE message is to be sent to different Subnetworks, then another GOOSE control block instance shall be used.

```
<xs:complexType name="tServerAt">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:attribute name="apName" type="tAccessPointName" use="required"/>
            <xs:attribute name="apUuid" type="tUUIDAttribute" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The *ServerAt* element has only the attribute apName and apUuid, which references the AccessPoint on the same IED, which hosts / defines the server's data model.

Since publication of IEC 61850-6:2009/AMD2, it is allowed to connect the AccessPoint of a ServerAt to the same SubNetwork of the corresponding Server or other ServerAt of the corresponding Server. Specific rules have been defined for backward/forward compatibility in Clause I.5.

The LN list allows a list of client LNs to be defined for a pure client access point. These client LNs can then be referenced to define the data flow between IEDs and to reserve control block instances, e.g. as reporting client to a report control block instance. For an example see IED A1KA1 in D.2.

Restrictions

- When the SCT is configuring reporting associated dataflow (ClientLN and ExtRef) when a Server and one or more ServerAt supporting report are connected to the same SubNetwork of the client, the client IED is responsible for determining which Access Point of the Server has to be used for the subscription with the ClientLN.apRef attribute.

9.3.3 The IED server

A communication server of the IED is described as follows:

```
<xs:complexType name="tServer">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Authentication">
                    <xs:complexType>
                        <xs:attributeGroup ref="agAuthentication"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="LDevice" type="tLDevice" maxOccurs="unbounded"/>
                <xs:element name="Association" type="tAssociation" minOccurs="0" maxOccurs="unbounded"/>
```

```
            </xs:sequence>
            <xs:attribute name="timeout" type="xs:unsignedInt" use="optional" default="30"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The IED server contains the elements *Authentication, LDevice* and *Association*. The attributes are defined as shown in Table 13.

Table 13 – Attributes of the IED server element

Attribute name	Description
timeout	Time out in seconds: if a started transaction (for example selection of a setting group) is not completed within this time, it is cancelled and reset
desc	A descriptive text

A server connection to a communication subnetwork is identified within the system by an access point. The access point identification in the communication system (address) is contained in the SCL communication section (see 9.4).

The mandatory *Authentication* element defines, in the case of a device description the authentication possibilities, in case of a device instantiated in a plant the method(s) to be used for authentication. If all attributes are missing, the default method is *none* (i.e. no authentication, meaning that the attribute *none* has the value *true*). The exact meaning of the other methods, especially weak and strong, is defined in the stack mappings (SCSMs).

```
<xs:attributeGroup name="agAuthentication">
    <xs:attribute name="none" type="xs:boolean" use="optional" default="true"/>
    <xs:attribute name="password" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="weak" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="strong" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="certificate" type="xs:boolean" use="optional" default="false"/>
</xs:attributeGroup>
```

The attributes of the *Authentication* element are defined in Table 14.

Table 14 – Attributes of the Authentication element

Attribute name	Description
none	No authentication
password	Is defined in the stack mappings (SCSMs)
weak	
strong	
certificate	

NOTE The *GOOSESecurity* and *SMVSecurity* elements of the access point are only allowed to be used if *certificate="true"* at the *Authentication* element.

9.3.4 The logical device

The *LDevice* element defines a logical device of the IED reachable via an access point. It shall contain at least the LN0, and may contain a preconfigured report, GSE and SMV definitions.

```
<xs:complexType name="tLDevice">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element ref="LN0"/>
                <xs:element ref="LN" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="AccessControl" type="tAccessControl" minOccurs="0"/>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInIED">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="inst" type="tLDInst" use="required"/>
            <xs:attribute name="ldName" type="tLDName" use="optional"/>
            <xs:attributeGroup ref="agUuid"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes of the LDevice element are defined in Table 15.

Table 15 – Attributes of the LDevice element

Attribute name	Description
inst	Identification of the LDevice within the IED. Its value cannot be the empty string. It is always used as key part for references to logical devices within the SCL file.
desc	The description text
ldName	The explicitly specified name of the logical device according to IEC 61850-7-1 and IEC 61850-7-2 within the communication. If missing, the default is the IED name concatenated with the inst value defined above
uuid	A unique identifier generated on creation of the logical device instance. Optional.
templateUuid	The template unique identifier of the logical when created by importing a template file with an identifier already defined. Optional.

Restrictions

- The LD *inst* shall be unique within the IED.
- The LD name built from *inst* and other parts as described in 8.5 shall be unique within each SCL file.
- The *ldName*, if specified, must be unique within each SubNetwork (even if distributed in several SCL files, e.g. SED files), and different to any default name of other LDs.
- The length of the attribute *inst* shall be at least one character.
- The length of the logical device name (either *ldName*, or IED name concatenated with *inst*) is restricted to 64 characters, and it is alphanumeric with only underscore (_) as additional character.
- The *uuid* is the instance identifier of the logical. It may be created at the same time the instance of the logical device is created. When it is not already defined in a previously imported ISD/IID file, then the SCT may create a uuid. When already defined in a previously imported ISD/IID then it is preserved. When it is present during the import of an ICD file, then the SCT conserves inside the templateUuid the uuid from the LDevice in the ICD file attribute, and creates a new uuid.

9.3.5 LN0 and other Logical Nodes

```
<xs:complexType name="tLN0">
    <xs:complexContent>
        <xs:extension base="tAnyLN">
            <xs:sequence>
                <xs:element name="GSEControl" type="tGSEControl" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="SampledValueControl" type="tSampledValueControl" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="SettingControl" type="tSettingControl" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="lnClass" type="tLNClassEnum" use="required" fixed="LLN0"/>
            <xs:attribute name="inst" type="xs:normalizedString" use="required" fixed=""/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The LN0 contains the following elements: *GSEControl* (see 9.3.10), *SampledValueControl* (see 9.3.11), and *SettingControl* (see 9.3.12. Furthermore, it inherits *ReportControl, Log* and *LogControl* from the base type tAnyLN, as well as the *DOI* and *Inputs* element.

The *Log* element indicates that the logical device, which is controlled by this LN0, contains a log, and its instance name ldInst can be used as log name within a log control block.

The LN contains the following elements: DataSet (see 9.3.7), ReportControl(see 9.3.8), LogControl (see 9.3.9), DOI (see 9.3.6) and Inputs (see 9.3.13). Further it contains Log element(s), if the LN contains one or more logs. The Log element has as its only attribute its name, which shall be unique within the LN. If this log name is missing, the default name value is the LD instance name (LD inst value).

The attributes of the LN are defined as shown in Table 16.

Table 16 – Attributes of the LN0 element

Attribute name	Description
desc	The description text for the logical node
lnType	The instantiable type definition of this logical node, reference to a LNodeType definition
lnClass	The LN class according to IEC 61850-7-x or other domain specific standards
inst	The LN instance number identifying this LN – an unsigned integer; leading zeros are not recommended, as they formally lead to another instance identification.
prefix	The LN prefix part
uuid	A unique identifier generated on creation of the logical node instance. Optional.
templateUuid	The template unique identifier of the logical node when created by importing a template file with an identifier already defined. Optional.

Restrictions

- The LN0 LN class is always LLN0, so no *inst* attribute is needed. For the referencing of links to LN0, *lnInst* shall be missing when attribute is optional or shall be empty when attribute is required, and *lnClass* shall be LLN0.

The Logical Node (type *tLN*) is described as follows:

```
<xs:complexType name="tLN">
    <xs:complexContent>
        <xs:extension base="tAnyLN">
            <xs:attribute name="lnClass" type="tLNClassEnum" use="required"/>
            <xs:attribute name="inst" type="tLNInst" use="required"/>
            <xs:attribute name="prefix" type="tPrefix" use="optional" default=""/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

tAnyLN, the super-type of both *tLN0* and *tLN*, is defined as follows:

```
<xs:complexType name="tAnyLN" abstract="true">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="DataSet" type="tDataSet" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="ReportControl" type="tReportControl" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="LogControl" type="tLogControl" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="DOI" type="tDOI" minOccurs="0" maxOccurs="unbounded">
                    <xs:unique name="uniqueSDI_DAIinDOI">
                        <xs:selector xpath="./scl:DAI|./scl:SDI"/>
                        <xs:field xpath="@name"/>
```

```
                <xs:field xpath="@ix"/>
            </xs:unique>
        </xs:element>
        <xs:element name="Inputs" type="tInputs" minOccurs="0"/>
        <xs:element name="Outputs" type="tOutputs" minOccurs="0"/>
        <xs:element name="Log" type="scl:tLog" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Labels" type="tLabels" minOccurs="0">
            <xs:unique name="uniqueLabelInIED">
                <xs:selector xpath="./scl:Label"/>
                <xs:field xpath="@id"/>
                <xs:field xpath="@lang"/>
            </xs:unique>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="lnType" type="tName" use="required"/>
    <xs:attributeGroup ref="agUuid"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

The LN contains the following elements: DataSet (see 9.3.7), ReportControl(see 9.3.8), LogControl (see 9.3.9), DOI (see 9.3.6) and Inputs (see 9.3.13). Further it contains Log element(s), if the LN contains one or more logs. The Log element has as its only attribute its name, which shall be unique within the LN. If this log name is missing, the default name value is the LD instance name (LD inst value).

The attributes of the LN are defined as shown in Table 17.

Table 17 – Attributes of the LN element

Attribute name	Description
desc	The description text for the logical node
lnType	The instantiable type definition of this logical node, reference to a LNodeType definition
lnClass	The LN class according to IEC 61850-7-x or other domain specific standards.
inst	The LN instance number identifying this LN – an unsigned integer; leading zeros are not recommended, as they formally lead to another instance identification
prefix	The LN prefix part
uuid	A unique identifier generated on creation of the logical node instance. Optional.
templateUuid	The template unique identifier of the logical node when created by importing a template file with an identifier already defined. Optional.

The optional DOI elements in an LN definition can be used to define special instance-related values for data objects and their attributes by using SDI elements for data object or attribute structure parts (if needed) and DAI elements per final attribute (see DOI definition in 9.3.6). The data objects and attributes referenced here shall however already be defined within the LNodeType definition of the LN, referenced with the LNType attribute of the LN. The DOI elements at this place for this instance shall NOT define new DOs or new attributes, which are not contained in the LNodeType. For example, the pulse length configuration parameter of a DPC CDC, specified with 100 ms in the LNodeType, is overwritten here with a value of 300 ms for this special DO. If the same value applies to several occurances, typical values can be defined within the DataTypeTemplate section. In this case an individual value can be used to override the typical value.

Restrictions

- The LN Name consisting of *prefix, lnClass* and *inst* shall be unique within the scope of the logical device, if a server is defined, or else within the scope of the IED.
- The *inst* attribute shall be a number with no more than 7 digits.
- The *prefix* follows the restrictions stated in IEC 61850-7-2.
- The *Log* element is only allowed in LLN0, and in some special LN classes explicitly defined in other parts of this standard or in other standards.
- The *uuid* is the instance identifier of the logical node (either LN or LN0. It may be created at the same time the instance of the logical node is created. When it is not already defined in a previously imported ISD/IID file, then the SCT may create a uuid. When already defined in a previously imported ISD/IID then it is preserved. When it is present during the import of an ICD file, then the SCT conserves inside the templateUuid the uuid from the logical node in the ICD file attribute, and creates a new uuid.

9.3.6 Data object (DOI) definition

```
<xs:complexType name="tDOI">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="SDI" type="tSDI">
                        <xs:unique name="uniqueSDI_DAIinSDI">
                            <xs:selector xpath="./scl:DAI|./scl:SDI"/>
                            <xs:field xpath="@name"/>
                            <xs:field xpath="@ix"/>
                        </xs:unique>
                    </xs:element>
                    <xs:element name="DAI" type="tDAI"/>
                </xs:choice>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInIED">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
```

```
                    <xs:field xpath="@lang"/>
                </xs:unique>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="name" type="tDataName" use="required"/>
        <xs:attribute name="ix" type="xs:unsignedInt" use="optional"/>
        <xs:attribute name="accessControl" type="xs:normalizedString" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The DOI is decribed by one of the following elements: *SDI* or *DAI*.

The attributes of the DOI are defined as shown in Table 18.

Table 18 – Attributes of the DOI element

Attribute name	Description
desc	The description text for the data
name	A standardized DO name for example from IEC 61850-7-4. It is the root name part as defined in the LNodeType definition. Its value must be unique at this level, i.e. there shall be at maximum one DOI element for the same data object.
ix	Index of a data element in case of an array type; shall not be used if DOI has no array type
accessControl	Access control definition for this data. The empty string (default) means that the higher-level access control definition applies. Possible values are SCSM dependent.

The DAI attribute within the DOI defines the attributes and the related values to be set. Again, all attributes shall also be contained in the LNodeType definition of this LN. Only those are repeated here, where some additional (attribute or element) values shall be set or individually overwritten.

```
<xs:complexType name="tDAI">
  <xs:complexContent>
    <xs:extension base="tUnNaming">
      <xs:sequence>
        <xs:element name="Val" type="tVal" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Labels" type="tLabels" minOccurs="0">
          <xs:unique name="uniqueLabelInIED">
            <xs:selector xpath="./scl:Label"/>
            <xs:field xpath="@id"/>
            <xs:field xpath="@lang"/>
          </xs:unique>
        </xs:element>
```

```
            </xs:sequence>
            <xs:attribute name="name" type="tAttributeNameEnum" use="required"/>
            <xs:attribute name="sAddr" use="optional">
                <xs:simpleType>
                    <xs:restriction base="xs:normalizedString">
                        <xs:maxLength value="255"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="valKind" type="tValKindEnum" use="optional"/>
            <xs:attribute name="ix" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="valImport" type="xs:boolean" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The DAI contains the elements *Val* (see 9.5.4).

The DAI allows the description of instance values for an IED. This can be used at the engineering stage by other IEDs/LNs which need to know configuration-related values, for example if they have no services to read the values, or if the IED does not support their reading. Alternatively it can be used by the IED itself to set these values, either to offer them via the communication protocol, or at least consider them in its internal functions.

The attributes of the DAI are defined as shown in Table 19.

Table 19 – Attributes of the DAI element

Attribute name	Description
desc	The description text for the DAI element
name	The name of the Data attribute whose value is given. It is the last name part in a structured attribute name.
sAddr	Short address of this Data attribute
valKind	The meaning of the value from the engineering phases. If missing, the *valKind* from the type definition applies for any attached value.
ix	Index of the DAI element in case of an array type
valImport	if true, an IED / IED configurator supports import of values modified by another tool, even if valKind=RO or valKind=Conf; the default value is defined in the DatatypeTemplate section. It is the responsibility of the IED configurator to assure value consistency and value allowance even if valImport is true.

The DAI element contains a subset of the DA attributes, and shall be used within an IED DOI specification if some instance specific attribute values are set or typical attribute values overwritten. The DAI name together with the index value, if supplied, shall be unique at the specified level.

The subset of data or data attributes are described as follows:

```
<xs:complexType name="tSDI">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="SDI" type="tSDI"/>
                    <xs:element name="DAI" type="tDAI"/>
                </xs:choice>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInIED">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="tAttributeNameEnum" use="required"/>
            <xs:attribute name="ix" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="sAddr" use="optional">
                <xs:simpleType>
                    <xs:restriction base="xs:normalizedString">
                        <xs:maxLength value="255"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The SDI element stands for a substructure name part, either from a DO (corresponding to SDO in LNodeType) or a DA substructure name, except the final (leaf) attribute name. The SDI element contains either the elements SDI for a further structure name part, or DAI for the final attribute element with the value(s).

The attributes of the SDI element are defined as shown in Table 20.

Table 20 – Attributes of the SDI element

Attribute name	Description
desc	A description text for the SDI part
name	Name of the SDI (structure part)
ix	Index of the SDI element in case of an array type
sAddr	The short address for the whole structure. Shall only used if no lower level attribute has an sAddr value at instance or type definition.

Restrictions for DAI and SDI

- The name shall begin with a lower-case letter, except SIUnit and the exceptions defined in IEC 61850-8-1.
- Dots (.) are not allowed within names, only alphanumeric characters
- the combination of name value and ix value shall be unique at each level (e.g. inside a DOI element)

Example:

The following example describes the value of a structured DO and an array DO as DOI

```
<DOI name="Volts">
    <SDI name="sVC">
        <DAI name="offset"><Val>0</Val></DAI>
        <DAI name="scaleFactor"><Val>200</Val></DAI>
    </SDI>
</DOI>
 <DOI name="TmASt" desc="Example of array value definition – function wise meaningless">
     <SDI name="crvPts" ix="0">
         <DAI name="xVal"><Val>3.2</Val></DAI>
         <DAI name="yVal"><Val>32.2</Val></DAI>
     </SDI>
     <SDI name="crvPts" ix="1">
         <DAI name="xVal"><Val>12.5</Val></DAI>
         <DAI name="yVal"><Val>22.1</Val></DAI>
     </SDI>
     <SDI name="crvPts" ix="2">
         <DAI name="xVal"><Val>102.5</Val></DAI>
         <DAI name="yVal"><Val>2.1</Val></DAI>
    </SDI>
 </DOI>
```

9.3.7 Data set definition

```
<xs:complexType name="tDataSet">
        <xs:extension base="tUnNaming">
        <xs:choice maxOccurs="unbounded">
            <xs:element name="FCDA" type="tFCDA"/>
        </xs:choice>
        <xs:attribute name="name" type="tDataSetName" use="required"/>
        <xs:attributeGroup ref="agUuid"/>
    </xs:extension>
</xs:complexType>
```

The DataSet contains a sequence of FCDA elements. The data set definition of the LN has the following attributes (see Table 21):

Table 21 – Attributes of the DataSet element

Attribute name	Description
name	A name identifying this data set in the LN where it is defined
desc	The description text for the data set
uuid	A unique identifier generated on creation of the data set instance. Optional.
templateUuid	The template unique identifier of the data set when created by importing a template file with an identifier already defined. Optional.

```
<xs:complexType name="tFCDA">
    <xs:attribute name="ldInst" type="tLDInst" use="optional"/>
    <xs:attribute name="prefix" type="tPrefix" use="optional" default=""/>
    <xs:attribute name="lnClass" type="tLNClassEnum" use="optional"/>
    <xs:attribute name="lnInst" type="tLNInst" use="optional"/>
    <xs:attribute name="doName" type="tName" use="optional"/>
    <xs:attribute name="daName" type="tName" use="optional"/>
    <xs:attribute name="fc" type="tFCEnum" use="required"/>
    <xs:attribute name="ix" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="lnUuid" type="tUUIDAttribute" use="optional"/>
</xs:complexType>
```

The FCDA element defines the name of a functionally constrained data or functionally constrained data attribute according to IEC 61850-7-2 of this Server to be contained in the data set. The element has the following attributes (see Table 22):

Table 22 – Attributes of the FCDA element

Attribute name	Description
ldInst	The LD where the DO resides; shall always be specified except for GSSE
prefix	Prefix identifying together with *lnInst* and *lnClass* the LN where the DO resides; optional, default value is the empty string
lnClass	LN class of the LN where the DO resides; shall always be specified except for GSSE DataLabel empty string
lnInst	Instance number of the LN where the DO resides; shall be specified except for LLN0
doName	A name identifying the DO (within the LN). A name standardized in IEC 61850-7-4. The doName attribute is mandatory if the dataset is used for any other service than the deprecated GSSE. For elements or parts of structured data object types, all name parts are contained, separated by dots (.), down to (but without) the level where the fc is defined. If an SDO array element is selected, the appropriate name part shall contain at its end before a possible dot the array element number in the form *(ArrayElementNumber)*.
daName	The attribute name – if missing, all attributes with functional characteristic given by *fc* are selected. For elements or parts of structured data types, all name parts are contained, separated by dots (.), starting at the level where the fc is defined. If an attribute's array element is selected, the appropriate attribute name part shall contain at its end before any separating dot the array element number in the form *(ArrayElementNumber)*.
fc	All attributes of this functional constraint are selected. Possible constraint values see IEC 61850-7-2 or the *fc* definition in 9.5 The fc shall match the definition of the CDC. In case a daName is defined, the fc shall match the functional constraint defined in the DOType. If daName is not defined, the fc shall indicate a functional constraint existing in the DOType defining the FCD, and this functional constraint shall be allowed to be added to a dataset, as per IEC 61850-7-2 definition.
ix	An index to select an array element in case that one of the data elements is an array. The ix value shall be identical to the ArrayElementNumber value in the doName or daName part.
lnUuid	The UUID of the LN or LN0 which is referenced by this FCDA. Optional.

The order of data within a message based on this data set definition shall be the FCDA order in the data set. If an FCDA specifies a set of attributes via *fc*, then the order of data values is specified by the data object definition and the attribute order in the corresponding LNs *DOType*. The attribute order inside the DatatypeTemplate section *DOType* and *DAType* elements shall follow the CDC definitions of IEC 61850-7-3.

Restrictions

- If *daName* contains a non empty value, then the *fc* value must be valid for the attribute (i.e. defined identically at the appropriate *LNodeType* definition), otherwise the SCL file processing shall be stopped with an error message.
- If all attributes of the FCDA (except *fc*) are missing or empty, then this corresponds to an empty string in a GSSE DataLabel definition (*fc* value should be ST) – in all other data sets, this is not allowed and ldInst, lnClass, lnInst, doName and fc are mandatory. Observe that GSSE is deprecated and only mentioned here for backwards compatibility.
- All control blocks, which reference a data set, shall be contained in the same LN as the data set definition. Therefore, the data set reference within all control blocks only contains the LN relative data set name (Name attribute at DataSet element), and not its full name (which also contains the LD name and LN name according to IEC 61850-7-2).
- The data set name length is syntactically restricted to 32 characters, which corresponds to a general name length specifiable in the Services element of 64 characters. Observe that the default name length of the Services element of 32 means that the allowed usable data set name length is even shorter, i.e. 20 characters.

- An FCDA shall not reference a data which cannot be seen online, i.e. if valKind is Conf or Spec for all DA represented by the FCD/FCDA
- A DataSet shall only contain data from its own Server and shall not include data from another Server from the same IED.

Data set example

```
<DataSet name="Example">
    <FCDA ldInst="C1" prefix="" lnInst="1" lnClass="CSWI" doName="Pos" fc="ST"
        lnUuid="8bbb34fc-872a-4696-a07f-fa8f53b7500f"/>
    <FCDA ldInst="C1" prefix="" lnInst="2" lnClass="CSWI" doName="Pos" fc="ST"
        lnUuid="58097b2b-a2e4-4476-94f5-e469ba7854f8"/>
    <FCDA ldInst="C1" prefix="" lnInst="1" lnClass="MMXU" doName="A" fc="MX"
        lnUuid="c383de97-c091-4e9f-8d35-7080ecb081fb"/>
    <FCDA ldInst="C1" prefix="" lnInst="1" lnClass="MMXU" doName="PhV.phsA" fc="MX" daName="cVal"
        lnUuid="c383de97-c091-4e9f-8d35-7080ecb081fb"/>
    <FCDA ldInst="C1" lnInst="1" lnClass="PVOC" doName=" TmASt " fc="SP" daName="crvPts(2).xVal" ix="2"
        lnUuid="d3d86b92-bea8-4050-8b54-50bc6ddaf47b"/>
    <FCDA ldInst="C1" lnInst="1" lnClass="MHAI" doName="HPhV.phsAHar(3)" fc="MX" daName="mag" ix="3"
        lnUuid="1ff47066-a356-4e39-b409-62af940e3f46"/>
</DataSet>
```

9.3.8 Report control block

A report control block definition of the LN is as follows:

```
<xs:complexType name="tReportControl">
    <xs:complexContent>
        <xs:extension base="tControlWithTriggerOpt">
            <xs:sequence>
                <xs:element name="OptFields">
                    <xs:complexType>
                        <xs:attributeGroup ref="agOptFields"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="RptEnabled" type="tRptEnabled" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="rptID" type="tMessageID" use="optional"/>
            <xs:attribute name="confRev" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="buffered" type="xs:boolean" use="optional" default="false"/>
            <xs:attribute name="bufTime" type="xs:unsignedInt" use="optional" default="0"/>
            <xs:attribute name="indexed" type="xs:boolean" use="optional" default="true"/>
```

```
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="tControlWithTriggerOpt" abstract="true">
    <xs:complexContent>
        <xs:extension base="tControl">
            <xs:sequence>
                <xs:element name="TrgOps" type="tTrgOps" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="intgPd" type="xs:unsignedInt" use="optional" default="0"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="tControl" abstract="true">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:attribute name="name" type="tCBName" use="required"/>
            <xs:attribute name="datSet" type="tDataSetName" use="optional"/>
            <xs:attributeGroup ref="agUuid"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The report control block (RCB) contains the elements:*TrgOps*, *OptFields* and *RptEnabled*.

The attributes given in Table 23 are used.

Table 23 – Attributes of the report control block element

Attribute name	Description
name	Name of the report control block. This name is relative to the LN hosting the RCB, and shall be unique within the LN
desc	The description text
datSet	The name of the data set to be sent by the report control block; datSet should only be missing within an ICD-File, or to indicate an unused control block. The referenced data set must be in the same LN as the control block.
intgPd	Integrity period in milliseconds – see IEC 61850-7-2. Only relevant if trigger option *period* is set to true
rptID	Identifier for the report control block, optional; if not used, its value shall be set to NULL (see IEC 61850-7-2)
confRev	The configuration revision number of this report control block. The value 0 is only allowed for a control block without data set reference. A reset by the system configurator is not allowed. It is recommended to increment this by 10 000 on each configuration change, to distinguish this from online changes leading to an increment of 1 only.

Attribute name	Description
buffered	Specifies if reports are buffered or not – see IEC 61850-7-2; default: false
bufTime	Buffer time – see IEC 61850-7-2; default: 0
indexed	If true, the report control block instance names are built from the supplied name, followed by an index number from 01 up to maximum 99. Default: true. The value false is not allowed for SCT created instances.
uuid	A unique identifier generated on creation of the report control block instance. Optional.
templateUuid	The template unique identifier of the report control clock when created by importing a template file with an identifier already defined. Optional.

Restrictions

- The *uuid* is the instance identifier of the report control block. It may be created at the same time the instance of the report control block is created. When it is not already defined in a previously imported ISD/IID file, then the SCT may create a uuid. When already defined in a previously imported ISD/IID then it is preserved. When it is present during the import of an ICD file, then the SCT conserves inside the templateUuid the uuid from the report control block in the ICD file attribute, and creates a new uuid.
- The *uuid* is shared between all instances when RptEnable.max is greater than 1.

The attributes of element *TrgOps* are defined as follows:

```
<xs:complexType name="tTrgOps">
    <xs:attribute name="dchg" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="qchg" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="dupd" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="period" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="gi" type="xs:boolean" use="optional" default="true"/>
</xs:complexType>
```

If an attribute is not given, its value (the corresponding trigger option) is false, meaning that the trigger option shall not be used. The only exception is the *gi* trigger option, which per default is true due to backwards compatibility reasons.

The element OptFields is defined as follows:

```
<xs:element name="OptFields">
    <xs:complexType>
        <xs:attributeGroup ref="agOptFields"/>
    </xs:complexType>
</xs:element>
<xs:attributeGroup name="agOptFields">
    <xs:attribute name="seqNum" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="timeStamp" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="dataSet" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="reasonCode" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="dataRef" type="xs:boolean" use="optional" default="false"/>
```

```
    <xs:attribute name="entryID" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="configRef" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="bufOvfl" type="xs:boolean" use="optional" default="true"/>
</xs:attributeGroup>
```

Setting one of the attributes to true means that the corresponding data shall be included in the report (see IEC 61850-7-2). The default value of attribute *bufOvfl* is true, i.e. it has only to be set if it shall be false. The attribute *segmentation* is deprecated, because it has no meaning, and is removed from this version. It shall however be accepted as input for backward compatibility and should not be used, i.e. ignored at input.

The element RptEnabled is defined as follows:

```
<xs:complexType name="tRptEnabled">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="ClientLN" type="tClientLN" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="max" use="optional" default="1">
                <xs:simpleType>
                    <xs:restriction base="xs:unsignedInt">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The RptEnabled element contains the list of client LNs which can enable the reporting and for which the appropriate control block instance shall be reserved.

The attributes given in Table 24 are used.

Table 24 – Attributes of the RptEnabled element

Attribute name	Description
desc	The description text
max	Defines the maximum number of report control blocks of this type, which are instantiated at configuration time in the LN (and then used online). The default value is 1. A missing *RptEnabled* element within an ICD file indicates that this value shall be set by the system configurator within the limits defined by the *ConfReportControl* and *DynAssociation* element's *max* attributes.

According to IEC 61850-7-2, a report control block is dedicated to at most one client at a time. This means that if *max* > 1 is given for *RptEnabled*, more than one report control block (RCB) of this type is instantiated in the IED. Observe that for all permanently used buffered control blocks, a ClientLN shall be preconfigured, and ResvTms set to -1, if it exists. For all other control block instances an existing ResvTms shall be set to 0. If ClientLNs are preconfigured for unbuffered RCBs, then the *Resv* (URCB Reservation is described in IEC 61850-7-2) attribute of the RCB shall be set to true additionally to the *RptEna* attribute (Report Enable is described in IEC 61850-7-2) in the IED. The URCName or BRCName of the control block as defined in IEC 61850-7-2 is built from the RCName attribute above by either using it directly if the attribute *indexed* is set to false, or (if *indexed*=true) followed by a two digit number between 01 and *max*. If *ClientLN*s are defined and the attribute *indexed* is set to *true* (which is the default value), the index (position) of the ClientLN in the list contained in the RptEnabled element is used as this number for this client (the first client relates to index 01). This means that a report control block definition in SCL has to be considered as a type, and not as an instance, which might have 99 instances for 99 clients. In case of buffered control blocks *indexed* may only be set to *false*, if only one instance of this type is possible, i.e. max=1.

The count of instances created by the device is independent from the indexed attribute. The attribute max is indicating the number of instances created by the device and the attribute indexed indicates if each instance name is created with the index number.

The ClientLN element defines the name of an LN in the system, which is a client to this report CB type.

```
<xs:complexType name="tClientLN">
    <xs:attributeGroup ref="agLNRef"/>
    <xs:attribute name="apRef" type="tAccessPointName" use="required"/>
</xs:complexType>

<xs:attributeGroup name="agLDRef">
    <xs:attributeGroup ref="scl:agDesc"/>
    <xs:attribute name="iedName" type="tIEDName" use="required"/>
    <xs:attribute name="ldInst" type="tLDInst" use="required"/>
</xs:attributeGroup>

<xs:attributeGroup name="agLNRef">
    <xs:attributeGroup ref="agLDRef"/>
    <xs:attribute name="prefix" type="xs:normalizedString" use="optional"/>
    <xs:attribute name="lnClass" type="tLNClassEnum" use="required"/>
    <xs:attribute name="lnInst" type="xs:normalizedString" use="required"/>
    <xs:attribute name="lnUuid" type="tUUIDAttribute" use="optional"/>
</xs:attributeGroup>
```

The attributes given in Table 25 are used.

Table 25 – Attributes of the ClientLN element

Attribute name	Description
iedName	The name of the IED where the LN resides
apRef	The name of the access point via which the IED shall be accessed.
ldInst	The instance identification of the LD where the LN resides
prefix	The LN prefix
lnClass	The LN class according to IEC 61850-7-4 or domain specific standards
lnInst	The instance id of this LN instance of below LN class in the IED
desc	optional descriptive text, e.g. about purpose of the client
lnUuid	The UUID of the LN or LN0 which is referenced by this ClientLN. Optional.

Observe that if the buffered control blocks support the online *ResvTms* attribute, that then on loading the SCD file the value of this attribute shall be set to -1 (reserved) for all instances allocated to clients, and to 0 (free) for all other instances.

ClientLN restrictions

- Both the iedName and ldInst (as specified in the attribute group agLDRef) shall be specified with non-zero length. If the reference is to an LN at a pure client access point, then the value of ldInst shall be LD0. Note that earlier SCL versions might use other ldInst values here.
- Only the prefix is optional in cases where the referenced LN instance has no prefix (prefix value = empty string). If it is defined, it must have a non zero length.
- lnInst is required. If the LLN0 is referenced, the value is the empty string.

Report control block restrictions

- The name of the report control block shall be unique within the LN. It contains only alphanumeric characters.
- The datSet attribute must contain a valid reference. If an unused control block is in the IED, then for this the datSet attribute must be left out completely. The data set is referenced by its LN relative name only, i.e. it shall reside in the same LN as the control block.
- The rptID can be missing, if the NULL value according to IEC 61850-7-2 is used. However, if the attribute is used, its value shall not be the empty string (however the empty string shall be accepted as input for backward compatibility).
- The confRev value 0 is only allowed for newly created control blocks without data set reference. For any change of the data set structure or the data set reference, inclusive removal of this, confRev shall be incremented.

RptEnabled restrictions

- When ReportControl *buffered*="false" and *indexed*="false", RptEnabled max shall be 1 or equal to DynAssociation max attribute.

Note that to identify a LN within the system, the IED-based designation is used within SCL, even if the communication level name is based on a separately supplied IdName. It is recommended that a tool assures that the defined client is really accessible across the defined communication system.

For pre-established associations, the AssociationId corresponding to the referenced LN can be found in the association definition section of this IED as defined in 9.3.14.

Example:

```
<ReportControl name="PosReport" rptID="E1Q1Switches" datSet="Positions" confRev="1"
               uuid="a0e0d63e-9209-4fb6-bea9-fa2663bd964c">
    <TrgOps dchg="true" qchg="true"/>
    <OptFields/>
    <RptEnabled max="5">
        <ClientLN iedName="A1KA1" apRef="S1" ldInst="none" lnInst="1" lnClass="IHMI"
                  lnUuid="7884c2da-3e95-44d6-b586-c27c24eb93bb"/>
    </RptEnabled>
</ReportControl>
```

The RptEnabled part defines that the Report control block type is valid for 5 (unbuffered; missing *buffered* attribute) RCBs with names PosReport01, PosReport02, up to PosReport05 (missing *indexed* attribute means *true*). The first one, PosReport01, is already reserved for the client A1KA1LD1/IHMI1. All reports are triggered with dchg and qchg, and the buffer time is 0. No OptFields are defined, i.e. only the mandatory information is included in the report.

9.3.9 Log control block

A log control block is defined by the following element:

```
<xs:complexType name="tLogControl">
    <xs:complexContent>
        <xs:extension base="tControlWithTriggerOpt">
            <xs:attribute name="ldInst" type="tLDInst" use="optional"/>
            <xs:attribute name="prefix" type="tPrefix" use="optional" default=""/>
            <xs:attribute name="lnClass" type="scl:tLNClassEnum" use="optional" default="LLN0"/>
            <xs:attribute name="lnInst" type="tLNInst" use="optional"/>
            <xs:attribute name="logName" type="tLogName" use="required"/>
            <xs:attribute name="logEna" type="xs:boolean" use="optional" default="true"/>
            <xs:attribute name="reasonCode" type="xs:boolean" use="optional" default="true"/>
            <xs:attribute name="bufTime" type="xs:unsignedInt" use="optional" default="0"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The meaning of the attributes is mostly identical to the appropriate control block attributes defined in IEC 61850-7-2. For those where it is completely identical the same attribute name is used.

The attributes of the log control block element are defined in Table 26.

Table 26 – Attributes of the log control block element

Attribute name	Description
name	the name of the log control block
desc	a description text
datSet	the name of the data set whose values shall be logged; datSet should only be missing within an ICD-File, or for an unused control block. The referenced data set must reside in the same LN as the control block.
intgPd	integrity scan period in milliseconds – see IEC 61850-7-2.
ldInst	The identification of the LD where the log resides; if missing, the same LD where this control block is placed.
prefix	Prefix of LN where the log resides; if missing, empty string
lnClass	Class of the LN where the log resides; if missing, LLN0
lnInst	Instance number of LN, where the log resides; missing for LLN0
logName	Relative name of the log within its hosting LN; name of the log element
logEna	TRUE enables immediate logging; FALSE prohibits logging until enabled online
reasonCode	If true, the reason code for the event trigger is also stored into the log – see IEC 61850-7-2
uuid	A unique identifier generated on creation of the log control block instance. Optional.
templateUuid	The template unique identifier of the log control block when created by importing a template file with an identifier already defined. Optional.

Restrictions

- The name of the log control block shall be unique within the LN.
- The datSet attribute shall contain a valid data set reference, or be missing completely. The data set reference is the LN relative name only, i.e. data set and control block shall reside in the same LN.
- The log reference shall point to a valid, defined log
- The *uuid* is the instance identifier of the log control block. It may be created at the same time the instance of the log control block is created. When it is not already defined in a previously imported ISD/IID file, then the SCT may create a uuid. When already defined in a previously imported ISD/IID then it is preserved. When it is present during the import of an ICD file, then the SCT conserves inside the templateUuid the uuid from the log control block in the ICD file attribute, and creates a new uuid.

The following extract of an SCL file shows a log control block example, which logs data from the data set Positions into the log C1 of the same logical device where this LCB is located, triggered by either data change or quality change.

```
<LogControl name="LogPos" datSet="Positions" logName="C1" uuid="3d14d0ea-f9e9-4db8-8584-548860ccf228">
    <TrgOps dchg="true" qchg="true"/>
</LogControl>
```

9.3.10 GSE control block

The following GSE control element is only allowed in the logical node LLN0.

```
<xs:complexType name="tGSEControl">
    <xs:complexContent>
        <xs:extension base="tControlWithIEDName">
            <xs:sequence>
                <xs:element name="Protocol" type="tProtocol" fixed="R-GOOSE" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="type" type="tGSEControlTypeEnum" use="optional" default="GOOSE"/>
            <xs:attribute name="appID" type="tMessageID" use="required"/>
            <xs:attribute name="fixedOffs" type="xs:boolean" use="optional" default="false"/>
            <xs:attribute name="securityEnable" type="scl:tPredefinedTypeOfSecurityEnum" use="optional" default="None"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>


<xs:complexType name="tControlWithIEDName">
    <xs:complexContent>
        <xs:extension base="tControl">
            <xs:sequence>
                <xs:element name="IEDName" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:simpleContent>
                            <xs:extension base="tIEDName">
                                <xs:attribute name="apRef" type="tAccessPointName" use="required"/>
                                <xs:attribute name="ldInst" type="tLDInst" use="optional"/>
                                <xs:attribute name="prefix" type="tPrefix" use="optional"/>
                                <xs:attribute name="lnClass" type="tLNClassEnum" use="optional"/>
                                <xs:attribute name="lnInst" type="tLNInst" use="optional"/>
                                <xs:attribute name="apUuid" type="tUUIDAttribute" use="optional"/>
                                <xs:attribute name="ldUuid" type="tUUIDAttribute" use="optional"/>
                                <xs:attribute name="lnUuid" type="tUUIDAttribute" use="optional"/>
                            </xs:extension>
                        </xs:simpleContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
```

```
            <xs:attribute name="confRev" type="xs:unsignedInt" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes given in Table 27 are used.

Table 27 – Attributes of the GSE control block element

Attribute name	Description
name	The name identifying this GOOSE control block
desc	A description text
datSet	The name of the data set to be sent by the GSE control block. For type=GSSE, the FCDA definitions in this data set shall be interpreted as DataLabels according to IEC 61850-7-2. The attribute datSet should only be missing within an ICD-File, or to indicate an unused control block. It resides in LLN0 like the control block
confRev	The configuration revision number of this control block. It is recommended to increment this by 10 000 on each configuration change, to distinguish this from online changes leading to an increment of 1 only
type	If the *type* is *GSSE (deprecated)*, then only single indication and double indication data types are allowed for the data items referenced in the data set, otherwise all data types are allowed. Note that on stack level each type might be mapped differently to message formats. The default type value is GOOSE
appID	A system wide unique identification of the application to which the GOOSE message belongs
fixedOffs	Default value false. If set to true it shows all receivers, that the values within the GOOSE message have fixed offset in the GOOSE message until a reconfiguration. This might mean for an MMS mapping that e.g. for integer values always the maximum size is used, although ASN.1 would allow a shorter coding.
securityEnabled	Default: None. Allows to configure the message security options per control block instance: Signature or SignatureAndEncryption. Only those indicated by the McSecurity element of the GSESetting are allowed. The security configured applies to non simulated and simulated GSE.
uuid	A unique identifier generated on creation of the GSE control block instance. Optional.
templateUuid	The template unique identifier of the GSE control block when created by importing a template file with an identifier already defined. Optional.

The GSE control block may optionally contain IED names for those IEDs which have to subscribe to the GSE data. Additionally to the IED name it is allowed to specify the destination in more detail down to the logical node level. For this purpose the following additional optional attributes can be used:

Table 28 – Attributes of the IEDName element

Attribute name	Description
apRef	The reference to the access point on the IED, via which the data shall flow. Mandatory.
ldInst	Identifies the destination LD in the IED. Optional.
prefix	Destination LN prefix. Optional
lnClass	Destination LN class, optional. If missing, no destination LN at all
lnInst	Destination LN instance number, optional. If missing, either no destination LN, or lnClass = LLN0.
apUuid	The UUID of the AccessPoint which is referenced by this IEDName. Optional, to be used when IEDName relates to AccessPoint only.
ldUuid	The UUID of the LDevice which is referenced by this IEDName. Optional, to be used when IEDName relates to Logical Device only.
lnUuid	The UUID of the LN or LN0 which is referenced by this IEDName. Optional, to be used when IEDName relates to LN.

Restrictions

- The GSE control block name shall be unique within the LLN0, i.e. the logical device.
- The datSet attribute must contain a valid data set reference, or be missing completely. A referenced data set shall reside in LLN0, like the control block.
- The confRev attribute is mandatory if the type is GOOSE (respective the type attribute is not specified).
- Different applications within the station shall have unique *appId* values. It is up to the project/system engineer to decide what an application is.
- The confRev value 0 is only allowed for newly created control blocks without data set reference. For any change of the data set structure or the data set reference, inclusive removal of this, confRev shall be incremented.
- The type GSSE shall no longer be used; it exists only for backwards compatibility.
- The *uuid* is the instance identifier of the GSE control block. It may be created at the same time the instance of the GSE control block is created. When it is not already defined in a previously imported ISD/IID file, then the SCT may create a uuid. When already defined in a previously imported ISD/IID then it is preserved. When it is present during the import of an ICD file, then the SCT conserves inside the templateUuid the uuid from the GSE control block in the ICD file attribute, and creates a new uuid.

The following SCL extract shows an example of a GOOSE control block definition:

```
<GSEControl name="ItlPositions" datSet="Positions" appID="Itl" uuid="e39667ca-e306-47e2-adf6-67c8cf56be60">
    <IEDName apRef="S1" apUuid="4c3fca73-77f5-4820-85b3-ebcfd03f90ed">E1Q2SB1</IEDName>
</GSEControl>
```

Its relative name within this LLN0 is *ItlPositions*, its message contents is defined by the data set *Positions*, and it shall be used for the *Itl* application.

9.3.11 Sampled value control block

The following sampled value control block element is only allowed in the logical node LLN0.

```
<xs:complexType name="tSampledValueControl">
    <xs:complexContent>
        <xs:extension base="tControlWithIEDName">
            <xs:sequence>
                <xs:element name="SmvOpts">
                    <xs:complexType>
                        <xs:attributeGroup ref="agSmvOpts"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="Protocol" type="tProtocol" fixed="R-SV" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="smvID" type="tMessageID" use="required"/>
            <xs:attribute name="multicast" type="xs:boolean" default="true"/>
            <xs:attribute name="smpRate" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="nofASDU" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="smpMod" type="tSmpMod" use="optional" default="SmpPerPeriod"/>
            <xs:attribute name="securityEnable" type="scl:tPredefinedTypeOfSecurityEnum" use="optional" default="None"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The sampled value control block contains the element SmvOpts, and as extension of the schema type *tControlWithIEDName* it optionally contains several IED references of IEDs which shall receive the messages – see Table 28.

The attributes given in Table 29 are used.

Table 29 – Attributes of the sampled value control block element

Attribute name	Description
name	A name identifying this SMV control block
desc	The description text
datSet	The name of the data set whose values shall be sent; datSet should only be missing within an ICD-File, or to indicate an unused control block. A referenced data set must reside in LLN0.
confRev	The configuration revision number of this control block; mandatory. It is recommended to increment it by 10 000 on any configuration change, to distinguish this from online configuration changes leading to an increment of 1 only
smvID	Multicast CB: the MsvID for the sampled value definition as defined in IEC 61850-7-2 Unicast CB: the UsvID as defined in IEC 61850-7-2
multicast	*false* indicates Unicast SMV services only meaning that smvID = UsvID
smpRate	Sample rate as defined in IEC 61850-7-2. If no smpMod is defined, in samples per period, else as stated by smpMod.
nofASDU	Number of ASDU (Application service data unit) – see IEC 61850-9-2
smpMod	The sampling mode as defined in IEC 61850-7-2; default: SmpPerPeriod; if supported by the IED, also SmpPerSec and SecPerSample can be choosen. In these cases smpRate defines the appropriate sample number per second, or seconds between samples.
securityEnabled	Default: None. Allows to configure the message security options per control block instance: Signature or SignatureAndEncryption. Only those indicated by the McSecurity element of the SMVSetting are allowed. The security configured applies to non simulated and simulated SMV.
uuid	A unique identifier generated on creation of the SV control block instance. Optional.
templateUuid	The template unique identifier of the SV control block when created by importing a template file with an identifier already defined. Optional.

Restrictions

- The attribute datSet must contain a valid data set reference, or be missing completely,
- If Multicast is FALSE, i.e. this is a Unicast control block, then following restrictions apply:
  - A maximum of one subscriber IED shall be assigned to the instance.
  - The *UsvCBName* defined in IEC 61850-7-2 shall be set directly to the defined name.
  - the *Resv* attribute of the CB as defined in IEC 61850-7-2 shall be initialized to TRUE.
- If Multicast is TRUE, then the *MsvCBName* defined in IEC 61850-7-2 shall be set directly to the defined name.

The following attributes can be set:

```
<xs:attributeGroup name="agSmvOpts">
    <xs:attribute name="refreshTime" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="sampleSynchronized" type="xs:boolean" use="optional" fixed="true"/>
    <xs:attribute name="sampleRate" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="dataSet" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="security" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="timestamp" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="synchSourceId" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="sampleMode" type="xs:boolean" use="optional" default="false"/>
```

```
</xs:attributeGroup>
```

The attributes of the Smv Options element are defined in Table 30.

Table 30 – Attributes of the Smv Options element

Attribute name	Description
refreshTime	The meaning of the options is described in IEC 61850-7-2. If any of the attributes is set to true, the appropriate values shall be included into the SMV telegram
sampleRate	
dataSet	The meaning of the options is described in IEC 61850-7-2. If the attribute is set to true, the dataset name shall be included into the SMV telegram
security	See IEC 61850-9-2 for description
synchSourceId	if true, the SV message contains the identity of the synchronizing master clock according to IEC 61850-9-3; default = false
timestamp	If true, the SV message contains the timestamp of the transmission time of the packet. This timestamp is used for synchrophasor.
sampleMode	The meaning of the options is described in IEC 61850-7-2. If the attribute is set to true, the sample mode shall be included into the SMV telegram

The following options (see Table 31) are deprecated and removed from the SCL syntax of this edition of IEC 61850-6, however they should be accepted as input for backward compatibility.

Table 31 – Deprecated Smv options

Attribute name	Description
dataRef	This value is no longer supported in SV-Telegrams. Therefore only the value false can be accepted
sampleSynchronized	This value is now always in the SV telegrams. The option is kept in the syntax for backward compatibility. Only value true can be accepted

Restrictions

- The SV control block name shall be unique within the LLN0, i.e. within the LDevice.
- The confRev attribute is mandatory for the SV control block.
- The datSet attribute must contain a valid data set reference, or be missing completely. The referenced data set shall reside in LLN0, like the control block.
- The confRev value 0 is only allowed for newly created control blocks without data set reference. For any change of the data set structure or the data set reference, inclusive removal of this, confRev shall be incremented.
- The *uuid* is the instance identifier of the SV control block. It may be created at the same time the instance of the SV control block is created. When it is not already defined in a previously imported ISD/IID file, then the SCT may create a uuid. When already defined in a previously imported ISD/IID then it is preserved. When it is present during the import of an ICD file, then the SCT conserves inside the templateUuid the uuid from the SV control block in the ICD file attribute, and creates a new uuid.

The following SCL extract shows the definition of an SV control block, which refers to data set smv. This data set defines the data contents of the SV message:

```
<SampledValueControl name="Volt" datSet="smv" smvID="11" smpRate="4800" nofASDU="5"
                     uuid="695bbe00-5fef-4088-8b06-5923e499c5fd">
    <IEDName apRef="S1" apUuid="513d767b-410a-46c6-9afa-f9cea591768e">D1Q1SB4</IEDName>
    <SmvOpts refreshTime="true" sampleSynchronized="true" sampleRate="true"/>
</SampledValueControl>
```

9.3.12 Setting control block

The following defines the definition for a setting group control block (SGCB). Note that the SGCB name, i.e. its name part within the LN0, is SGCB according to IEC 61850-7-2. Therefore, only one SGCB is allowed per LN0.

```
<xs:complexType name="tSettingControl">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:attribute name="numOfSGs" use="required">
                <xs:simpleType>
                    <xs:restriction base="xs:unsignedInt">
                        <xs:minInclusive value="1"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="actSG" use="optional" default="1">
                <xs:simpleType>
                    <xs:restriction base="xs:unsignedInt">
                        <xs:minInclusive value="1"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
      <xs:attribute name="resvTms" type="xs:unsignedShort" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes are identical to those of the setting group control block in IEC 61850-7-2.

The attributes of the setting control block element are defined in Table 32.

Table 32 – Attributes of the setting control block element

Attribute name	Description
desc	The description text
numOfSGs	The number of setting groups available. The value shall be > 0.
actSG	The number of the setting group to be activated when loading the configuration. The default value is 1. Any SCL value shall be > 0.
resvTms	The time in seconds the SGCB stays reserved for editing. After this time the IED automatically closes an edit session, if the client has not closed it or not confirmed any changes (see IEC 61850-7-2). If this function is not supported, the attribute shall be missing. The appropriate IED capability (Table 11) defines if a system tool can modify any value supplied by the IED tool.

9.3.13 Binding to external signals

The input section element *Inputs* defines all external signals, i.e. signals sent from other LNs mostly on other IEDs, which are needed by the LN application to fulfill its function. The section allows also the binding of the signal to an IED internal address *intAddr*.

The *ExtRef* elements of *Inputs* sections can principally be used for the following purposes:

1) Specifying expected inputs (input templates) of logical devices or logical nodes from the view of the IED configuration tool. The intAddr attribute shall be set to a non empty value. The pDO might specify the needed input CDC via a standardized DO name, additional pDA values might indicate the expected data type of an attribute and the pLN might document the expected function providing the input. pServT might provide the expected service type for this input value. These attribute values, if supplied by the IED configurator, shall not be modified by the system configurator. A system configurator binding to these input templates must assure that the CDC specified by pDO and, if given, the attribute (base) type specified by pDA are met. For a final engineered data flow also the pServT, if given, shall be met by the serviceType value.
2) Specifying expected external signal inputs to a logical device or logical node from the system engineer (system configuration tool) view. Beneath the external reference in this case the serviceType might be set. The intAddr attribute, all source control block related attributes and all pXX attributes are missing.
3) Documenting the actual incoming signals to an IED / logical device / logical node based on the configured data flow. In this case beneath the reference to the source control block also the service type is given.
4) Documenting the binding of incoming external signals to IED internal addresses for those signals actually used by the IED. Here the intAddr value after binding of the external signal is also specified.

The engineering process allows IED tools that need the system configuration tool to perform a binding of predefined IED input signals to an external signal source. In this case the IED must have the attribute *noIctBinding* of the *ClientServices* capability element set to 'true'.

IED tools that are not restricted here (noIctBinding = 'false') may also provide predefined IED input signal templates, but must accept also unbound external signals for binding by the IED tool.

Some example use cases can be found in Annex H.

```
<xs:complexType name="tInputs">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="ExtRef" type="tExtRef" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Each ExtRef element references one external item, either at DO or at DA level. If intAddr is needed, it has to be used appropriately to this level. This means that for a DO level usage it might contain a mapping of several attributes.

```
<xs:complexType name="tDORef" abstract="true">
    <xs:complexContent>
        <xs:extension base="tBaseElement">
            <xs:attributeGroup ref="scl:agDesc"/>
            <xs:attribute name="iedName" type="tIEDNameOrRelative" use="optional"/>
            <xs:attribute name="ldInst" type="tLDInst" use="optional"/>
            <xs:attribute name="prefix" type="tPrefix" use="optional"/>
            <xs:attribute name="lnClass" type="tLNClassEnum" use="optional"/>
            <xs:attribute name="lnInst" type="tLNInst" use="optional"/>
            <xs:attribute name="doName" type="tFullDOName" use="optional"/>
            <xs:attribute name="pLN" type="tLNClassEnum" use="optional"/>
            <xs:attribute name="pDO" type="tFullDOName" use="optional"/>
            <xs:attributeGroup ref="agUuid"/>
            <xs:attribute name="lnUuid" type="tUUIDAttribute" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:complexType name="tExtRef">
    <xs:complexContent>
        <xs:extension base="tDORef">
            <xs:attribute name="daName" type="tFullAttributeName" use="optional"/>
            <xs:attribute name="intAddr" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="serviceType" type="tServiceType" use="optional"/>
            <xs:attribute name="srcLDInst" type="tLDInst" use="optional"/>
            <xs:attribute name="srcPrefix" type="tPrefix" use="optional"/>
            <xs:attribute name="srcLNClass" type="tLNClassEnum" use="optional"/>
            <xs:attribute name="srcLNInst" type="tLNInst" use="optional"/>
            <xs:attribute name="srcCBName" type="tCBName" use="optional"/>
            <xs:attribute name="pServT" type="tServiceType" use="optional"/>
            <xs:attribute name="pDA" type="tFullAttributeName" use="optional"/>
            <xs:attribute name="srcCBUuid" type="tUUIDAttribute" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes shown in Table 33 are used.

Table 33 – Attributes of the Input/ExtRef element

Attribute name	Description
iedName	The name of the IED from where the input comes. For IED internal references the value @ may be used.
ldInst	The LD instance name from where the input comes.
prefix	The LN prefix
lnClass	The LN class according to IEC 61850-7-x. Used to indicate the lnClass of the concrete binding part within a SCD.
lnInst	The instance id of this LN instance of above LN class in the IED; missing for a reference in LLN0. For backwards compatibility also the empty string shall be accepted for LLN0.
doName	A name identifying the DO (within the LN).In case of structured DO, the name parts are concatenated by dots (.).Used to indicate the DO (within the LN) of the concrete binding part within a SCD.
daName	The attribute designating the input. The IED tool should use an empty value if it has some default binding (intAddr) for all process input attributes of a DO (*fc* = ST or MX), especially for t and q. If the attribute belongs to a data type structure, then the structure name parts shall be separated by dots (.)
intAddr	The internal address to which the input is bound. Only the IED tool of the concerned IED shall use the value. All other tools shall preserve it unchanged.
desc	A free description / text. Can e.g. be used at system engineering time to tell the IED engineer the purpose of this incoming data.
serviceType	Optional, values: Poll, Report, GOOSE, SMV, Used to indicate the used service if the data flow is configured.
srcLDInst	The LD inst of the source control block – if missing, same as ldInst above
srcPrefix	The prefix of the LN instance, where the source control block resides; if missing, no prefix
srcLNClass	The LN class of the LN where the source control block resides; if missing, LLN0
srcLNInst	The LN instance number of the LN where the source control block resides – if missing, no instance number exists (LLN0)
srcCBName	The source CB name; if missing, then all other srcXX attributes shall also be missing, i.e. no source control block is given. Shall be supplied if derived from configured data flow.
pDO	a preconfigured DO name to indicate an expected DO name and CDC. Any binding must match the CDC.
pLN	a preconfigured LN class indicating an expected LN class containing the DO indicated by pDO
pDA	a preconfigured data attribute indicating the expected attribute. If configured, any bound attribute must match the data type specified by specified pDO CDC and pDA attribute value
pServT	a preconfigured service type indicating an expected service type. if configured, serviceType must match its value.
uuid	A unique identifier generated on creation of the ExtRef instance. Optional.
templateUuid	The template unique identifier of the ExtRef when created by importing a template file with an identifier already defined. Optional.
lnUuid	The UUID of the LN or LN0 which is referenced by this ExtRef. Optional.
srcCBUuid	The UUID of the source Control Block. Optional.

Restrictions

- The *uuid* is the instance identifier of the external reference. It may be created at the same time the instance of the external reference is created. When it is not already defined in a previously imported ISD/IID file, then the SCT may create a uuid. When already defined in a previously imported ISD/IID then it is preserved. When it is present during the import of an ICD file, then the SCT conserves inside the templateUuid the uuid from the external reference in the ICD file attribute, and creates a new uuid.

If attribute *intAddr* is specified and *iedName* as well as *ldInst* are missing, this allows an IED configuration tool to specify the available internal addresses for later binding to external references. In this case any values at *pDO* and possibly *pDA* allow to specify the CDC respective attribute type expected for this *intAddr*. Together with *lnClass* this allows to specify a functionally expected input. The *desc* attribute allows to give it additional informal meaning, the *serviceType* attribute the intended service quality. If at least the *pDO* attribute is specified, then a missing *pDA* attribute includes at least all the operational value attribute(s) of the DO, i.e. stVal, mag, t, q etc. It might allow binding of additional attributes supported by the service type. In this case, *intAddr* can also specify the addresses of all operational attributes in some IED tool specific way. So, both usages are supported additionally to a full link specification: *intAddr* with missing *iedName* and *ldInst* e.g. within an ICD or IID file, or only an external reference by means of iedName/ldInst/prefix/lnClass/lnInst but without *intAddr* coming from an SCD file. The *desc* attribute can be used to indicate the purpose of this link, dependent on the usage above.

The system configuration tool can create and remove ExtRef definitions without intAddr value and bind external references to an *intAddr* supplied via an ExtRef template from the IED tool, or remove the binding from it. It is not allowed to change intAddr values or values of the pXxx attributes, nor is it allowed to remove ExtRefs with a defined intAddr. The IED tool can bind external references to internal addresses, and supply templates with *intAddr* values, which should contain at least additionally the *pDO* to indicate the expected CDC of an external signal. The ExtRef added by a system configuration tool which is not bound to an internal address by the IED tool shall remain in the resulting IID to let the system tool know that it is not used by the device. Some use cases are described in Annex H.

If the same input data can be received by the IED by different communication services (for example by report and by GOOSE), it is up to the project engineer or the IED respective its tool implementation to decide which one shall be taken, except an IED supplied ExtRef template exists which specifies already the needed serviceType. Any decision can be documented by means of the srcXX attributes, which allow to define the source control block for this input data. If *serviceType* is set to *Poll*, then no source control block shall be specified; this means that the client shall / will poll the input data by means of read requests.

Observe that the *serviceType* as well as the srcXx attributes are syntactically optional. For an SCD with completed data flow engineering they shall be supplied consistently with the data flow configured at the source.

Table 51 summarizes the usage of ExtRef attributes for the above defined purposes.

Table 51 – Usage of ExtRef attributes in different use cases

ExtRef attributes	Purpose use cases			
	ICD/IID – Specifying the later binding	SSD – SCD – Specifying expected external signal between specification and/or real device	SCD – Documenting the signals participating to the system dataflow	IID/SCD Documenting the complete binding of signals in the device
iedName	F	M	M	M
ldInst	F	M	M	M
prefix	F	MD	MD	MD
lnClass	F	M	M	M
lnInst	F	MD	MD	MD
doName	F	M	M	M
daName	F	O	O	O
intAddr	M	na	C	M
pLN	O	na	C	C
pDO	M – indirect specification of expected type on reception in combination with optional pDA	na	C	C
pDA	O – in combination with pDO indirect specification of expected basic type on reception	na	C	C
pServT	O – in combination with pDO specification of expected service type	na	C	C
desc	M	O	O	O
serviceType	F	O	M	M
srcLDInst	F	na	MCD	MCD
srcPrefix	F	na	MCD	MCD
srcLNClass	F	na	MCD	MCD
srcLNInst	F	na	MCD	MCD
srcCBName	F	na	MC	MC

na not applicable.
F Forbidden; shall not be defined;
MD Mandatory; can be missing if default applies.
MC Mandatory, if the serviceType value is NOT 'Poll'.
MCD Mandatory if srcCBName is given and the default does not apply.
C after SCT binding as supplied in ICD/IID file, else missing (na).
O Optional
M Mandatory.

The Inputs / ExtRef feature could also be used to describe the IED internal signal connections, i.e. the external reference is an IED internal signal. it is up to the IED tool if it supports this option either as documentation or even as manufacturer independent IED internal engineering feature. If such a binding is used, the ExtRef.serviceType shall be missing.

9.3.15 Binding to external controls

The output section element *Outputs* defines all external controls, i.e. controls sent to other LNs mostly on other IEDs, which are needed by the client LN application to fulfil its function. The section allows also the binding of the control to an IED internal address *intAddr*.

The *ExtCtrl* elements of *Outputs* sections can principally be used for the following purposes:

1) Specifying controls which have to be performed by a client function (output templates) from the view of the IED configuration tool. The intAddr attribute shall be set to a non-empty value. The pDO might specify the needed input CDC via a standardized DO name. These attribute values, if supplied by the IED configurator, shall not be modified by the system configurator. A system configurator binding to these output templates must assure that the CDC specified by pDO is met.
2) Documenting the binding of outgoing external controls to client IED internal addresses for those controls actually performed by the IED. Here the intAddr value after binding of the external signal is also specified.

The engineering process allows IED tools that need the system configuration tool to perform a binding of predefined IED input signals to an external signal source. In this case the IED must have the attribute *noIctBinding* of the *ClientServices* capability element set to 'true'.

Client IED tools shall provide predefined IED output controls templates, and the engineering process allows the system configuration tool to perform a binding of predefined client IED output controls to an external control source. The system configuration tool does not allow to create additional output controls.

```
<xs:complexType name="tOutputs">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="ExtCtrl" type="tExtCtrl" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Each ExtCtrl element references one external item at DO level. If intAddr is needed, it has to be used appropriately to this level.

```
<xs:complexType name="tExtCtrl">
    <xs:complexContent>
        <xs:extension base="tDORef">
            <xs:attribute name="apRef" type="tAccessPointName" use="optional"/>
            <xs:attribute name="intAddr" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="checkSynchrocheck" type="tExtControlCheck" use="optional"/>
            <xs:attribute name="checkInterlock" type="tExtControlCheck" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes shown in Table 52 are used.

Table 52 – Attributes of the Output/ExtCtrl element

Attribute name	Description
iedName	The name of the IED where the output control is performed. For IED internal references the value @ may be used.
apRef	The name of the access point used to perform the output control.
ldInst	The LD instance name from where the output control is performed.
Prefix	The LN prefix
lnClass	The LN class according to IEC 61850-7-x. Used to indicate the lnClass of the concrete binding part within a SCD.
lnInst	The instance id of this LN instance of above LN class in the IED; missing for a reference in LLN0.
doName	A name identifying the controllable DO (within the LN). In case of structured DO, the name parts are concatenated by dots (.). Used to indicate the DO (within the LN) of the concrete binding part within a SCD.
intAddr	The internal address to which the output is bound. Only the IED tool of the concerned IED shall use the value. All other tools shall preserve it unchanged.
desc	A free description / text. Can e.g. be used at system engineering time to tell the IED engineer the purpose of this outgoing data
uuid	A unique identifier generated on creation of the ExtCtrl instance. Optional.
templateUuid	The template unique identifier of the ExtCtrl when created by importing a template file with an identifier already defined. Optional.
lnUuid	The UUID of the LN or LN0 which is referenced by this ExtCtrl. Optional.
checkSynchrocheck	Indicates how the client IED shall perform the synchrocheck check. Could take three values: true, false or conserve. Conserve is used for cascaded controls (i.e. by a gateway) to indicates that the check shall be performed as per the original request. Optional, default is true.
checkInterlock	Indicates how the client IED shall perform the interlock check. Same as checkSynchrocheck. Optional, default is true.

Restrictions

- The *uuid* is the instance identifier of the external control. It may be created at the same time the instance of the external control is created. When it is not already defined in a previously imported ISD/IID file, then the SCT may create a uuid. When already defined in a previously imported ISD/IID then it is preserved. When it is present during the import of an ICD file, then the SCT conserves inside the templateUuid the uuid from the external control in the ICD file attribute, and creates a new uuid.

Management of Outputs/ExtCtrl during engineering shall be the same as Inputs/ExtRef, except that ExtCtrl cannot be created by an SCT.

9.3.14 Associations

```
<xs:complexType name="tAccessControl" mixed="true">
    <xs:complexContent>
        <xs:extension base="tAnyContentFromOtherNamespace"/>
    </xs:complexContent>
</xs:complexType>
```

An access control definition. Meaning and eventual refinement of the definition are stack (SCSM)-specific issues.

Each association definition defines one pre-configured association between this server and a client logical node. Two kinds of pre-configuration are possible. *Predefined* means that this association is defined, but not yet opened, the client has to open it. *Pre-established* means that the association is defined and considered to be open directly after IED start up.

```
<xs:complexType name="tAssociation">
    <xs:attribute name="apRef" type="tAccessPointName" use="required"/>
    <xs:attribute name="kind" type="tAssociationKindEnum" use="required"/>
    <xs:attribute name="associationID" type="tName" use="optional" />
    <xs:attribute name="initiator" type="tAssociationInitiator" use="optional" default="client"/>
    <xs:attributeGroup ref="agLNRef"/>
</xs:complexType>
```

The attributes shown in Table 34 are used.

Table 34 – Attributes of the association element

Attribute name	Description
kind	The kind of pre-configured association, pre-established or predefined
associationID	The identification of a pre-configured association (otherwise missing)
iedName	The reference identifying the IED on which the client resides
apRef	The name of the access point via which the IED shall be accessed.
initiator	The initiator of the association, client or server. By default it's client. To select server, both client and server shall support server initiator.
ldInst	The reference to the client logical device
lnClass	The class of the client LN
prefix	The LN prefix
lnInst	The instance number of the client LN
lnUuid	The UUID of the LN or LN0 which is referenced by this Association. Optional.
apRef	The name of the access point via which the IED shall be accessed. Mandatory.
initiator	The initiator of the association, client or server. By default its client. To select server, both client and server shall support server initiator.
lnUuid	The UUID of the LN or LN0 which is referenced by this Association. Optional.

An empty association Id as given by the default value means that the association Id is not yet defined. For a completed SCL file and a pre-established association, the association Id shall be set, so that the client LNs and the server can verify it correctly. The same client may use the same association to different LNs on the same server. Uniqueness requirements as well as value range of the association Id (for example a 32 bit integer, unique at the server, or at server IED and client Id, or system wide) are set up in the SCSMs.

Restrictions

- The association ID shall be unique within the Server.
- The length of the association ID shall be at least one.

9.4 Communication system description

9.4.1 General

This clause describes the direct communication connection possibilities between logical nodes by means of logical busses (SubNetworks) and IED access points. The IED sections already describe which LDs and LNs are reachable across a certain access point. The communication section now describes which IED access points are connected to a common subnetwork. This is done in a way that reflects the hierarchical name structure within the IED, which is based on IED relative names for access points, LDs and LNs.

The UML diagram shown in Figure 22 gives an overview of the Communication section.

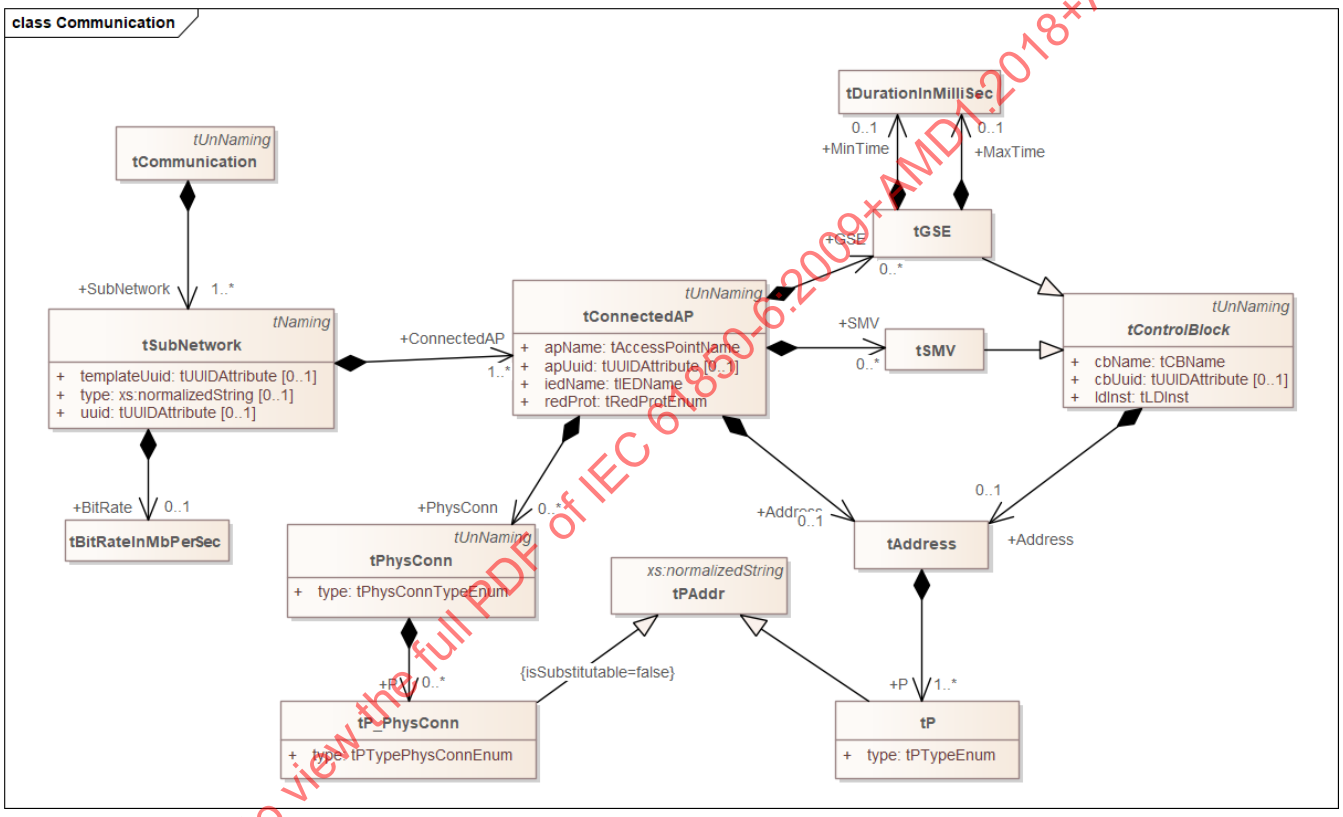


Figure 22 – UML diagram overview of the Communication section

The formal XML schema definition is as follows:

```
<xs:element name="Communication" type="tCommunication">
    <xs:unique name="uniqueSubNetwork">
        <xs:selector xpath="./scl:SubNetwork"/>
        <xs:field xpath="@name"/>
    </xs:unique>
</xs:element>

<xs:complexType name="tCommunication">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
```

```
            <xs:sequence>
                <xs:element name="SubNetwork" type="tSubNetwork" maxOccurs="unbounded">
                    <xs:unique name="uniqueConnectedAP">
                        <xs:selector xpath="./scl:ConnectedAP"/>
                        <xs:field xpath="@iedName"/>
                        <xs:field xpath="@apName"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The Communication section might optionally contain Text and Private sections (derivation from tUnNaming). The names of the SubNetworks shall be unique.

9.4.2 Subnetwork definition

A SubNetwork definition contains all access points which can (logically) communicate with the SubNetwork protocol and without the intervening router. Observe that a subnetwork defines a logical connection with a certain protocol. Different subnetworks with different protocols might run on the same physical communication network.

```
<xs:complexType name="tSubNetwork">
    <xs:complexContent>
        <xs:extension base="tNaming">
            <xs:sequence>
                <xs:element name="BitRate" type="tBitRateInMbPerSec" minOccurs="0"/>
                <xs:element name="ConnectedAP" type="tConnectedAP" maxOccurs="unbounded">
                    <xs:unique name="uniqueGSEinConnectedAP">
                        <xs:selector xpath="./scl:GSE"/>
                        <xs:field xpath="@cbName"/>
                        <xs:field xpath="@ldInst"/>
                    </xs:unique>
                    <xs:unique name="uniqueSMVinConnectedAP">
                        <xs:selector xpath="./scl:SMV"/>
                        <xs:field xpath="@cbName"/>
                        <xs:field xpath="@ldInst"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
```

```
            <xs:attribute name="type" type="tSubnetType" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes of a Subnetwork are defined as shown in Table 35.

Table 35 – Attributes of the Subnetwork element

Attribute	Description
name	A name identifying this bus; unique within this SCL file
desc	Some descriptive text to this SubNetwork
type	The SubNetwork protocol type; protocol types are defined by the SCSMs. In the examples, 8-MMS is used for the protocol defined in IEC 61850-8-1; IP should be used for all IP based protocols except those explicitly standardized. PHYSICAL should be used, if only physical connections shall be modeled, e.g. at a hub. The protocol type can be anything but has also a list of predefined values for IEC 61850-8-1 (8-MMS) and IEC 61850-8-2 (8-XMPP).
uuid	A unique identifier generated on creation of the Subnetwork in the SCD. Optional.
templateUuid	A unique identifier for a Subnetwork used by an IED template in the ICD/IID. Optional.

Protocol types are defined in the stack mappings (SCSM), IEC 61850-8-1, IEC 61850-8-2 and IEC 61850-9-2 for this standard series. Those of IEC 61850-8-1 start with "8-" and those of IEC 61850-9-2 with "9-" (except if they are identical). The protocol of IEC 61850-8-1 and IEC 61850-9-2 is for 8-MMS, and IEC 61850-8-2 is for 8-XMPP. Additionally, the type IP is predefined for all IP based protocols except those specifically standardized, to allow unique IP address checking across all protocols (subnetworks) on the same (physical) network. Although the type attribute is syntactically optional, it shall be used inside an SCD file.

The Subnetwork contains an optional BitRate element defining the bit rate in Mbit/s, and a list of IED access points by which these IEDs are connected to a SubNetwork with access points. It inherits Private and Text elements from tUnNaming.

```
<xs:complexType name="tConnectedAP">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Address" type="tAddress" minOccurs="0"/>
                <xs:element name="GSE" type="tGSE" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="SMV" type="tSMV" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="PhysConn" type="tPhysConn" minOccurs="0" maxOccurs="unbounded">
                    <xs:unique name="uniquePTypeInPhysConn">
                        <xs:selector xpath="./scl:P"/>
                        <xs:field xpath="@type"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
```

```
            <xs:attribute name="iedName" type="tIEDName" use="required"/>
            <xs:attribute name="apName" type="tAccessPointName" use="required"/>
            <xs:attribute name="redProt" type="scl:tRedProtEnum" use="optional"/>
            <xs:attribute name="apUuid" type="tUUIDAttribute" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The ConnectedAP is the IED access point connected to this SubNetwork.

It has the attributes shown in Table 36.

Table 36 – Attributes of the ConnectedAP element

Attribute	Description
iedName	a name identifying the IED
apName	a name identifying this access point within the IED
desc	some descriptive text for this access point at this subnetwork
redProt	The redundancy protocol used at this access point: allowed values are hsr, prp. rstp, none; no default value, i.e. value is not known if the attribute is missing. The allowed values are restricted by the IED capabilities (Services/RedProt).
apUuid	The UUID of the access point which is referenced by this ConnectedAP. Optional.

For further restrictions see IEC 61850-7-2:2003, Table 1.

Each connected access point optionally has one server-related address, and additional address information for real time communication-related control blocks such as GSE control and SMV control. If all three are missing, it describes only the Subnetwork connection topology, for example for communication performance studies. For a complete SCD file, either the server address or at least one control block address shall be specified. Further each ConnectedAP element shall reference an IED access point existing in the IED section.

The optional element *PhysConn* describes one or more physical connections to this access point.

9.4.3 Address definition

The *Address* element contains the address parameters of this access point at this bus for at least one parameter. The different parameters are defined within the contained P elements. The *type* attribute of P identifies the meaning of the value. The meaning of the P parameters depend on the subnetwork protocol type and therefore has to be specified in the appropriate SCSM. Those used for IEC 61850-8-1 and IEC 61850-9-2 are contained in the *type* enumeration type tPTypeEnum. For an explanation, see the appropriate standard parts.

```
<xs:complexType name="tAddress">
    <xs:sequence>
        <xs:element name="P" type="tP" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
```

The access point address shall be filled with a unique value at least for server type access points to get a complete SCD description.

```
<xs:complexType name="tP">
    <xs:simpleContent>
        <xs:extension base="tPAddr">
            <xs:attribute name="type" type="tPTypeEnum" use="required"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

tPAddr is a (non-empty) string containing no special characters such as LF, CR, or Tab. The pre-defined values for *tPTypeEnum* are as defined in IEC 61850-8-1. Custom-defined address types are also allowed (see below).

In order to be able to provide better validation of the address content by an XML parser, *tP* has been restricted (in the XML Schema sense) for each of these pre-defined address types. These type restrictions are named "*tP_*" followed by the address type as in *tPTypeEnum*. To use these restrictions, the *xsi:type* attribute must be given in the *P* element. Thus, there are two ways to provide such an address. For instance, for an IP address, both of the following formulations are equivalent from a syntactical and semantical point of view:

```
<P type="IP">10.0.0.11</P>
<P type="IP" xsi:type="tP_IP">10.0.0.11</P>
```

The advantage of the second, which uses the restriction type of *tP*, is that the address value (here "10.0.0.11") can also be validated by an XML parser. Using the first formulation, an address value of "abc" would be considered as perfectly valid, while the second formulation expects a value of the form "ddd.ddd.ddd.ddd", where each d corresponds to a digit.

Even if the restricted type is used, the (correct) address type must be specified.

When a P element is created with a standardized type, the corresponding *xsi:type* shall be provided in the element definition. This applies to the ICT which create P elements in ICD/IID, so it can be used by the SCT to validate the provided value, and this applies to the SCT when it creates the missing P elements (typically the IP addresses).

An SCT is required to keep *xsi:type* attribute provided in an ICD/IID and give them back in SCD.

Restrictions

- Extensions of the P *type* enumeration type *tPTypeEnum* shall start with a capital letter, and contain only alphanumeric characters and dashes(-),

9.4.4 GSE address definition

All control block address information is based on the abstract tControlBlock type. It provides the Address element for stating the control block-related address parameters, and the reference to the control block within the IED by means of the *ldInst* and *cbName* attributes. Since GSE as well as SMV control blocks shall be located within LLN0, this is sufficient.

```
<xs:complexType name="tControlBlock" abstract="true">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
```

```
            <xs:sequence>
                <xs:element name="Address" type="tAddress" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="ldInst" type="tLDInst" use="required"/>
            <xs:attribute name="cbName" type="tCBName" use="required"/>
            <xs:attribute name="cbUuid" type="tUUIDAttribute" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The GSE element defines the address for a GSE control block in this IED.

```
<xs:complexType name="tGSE">
    <xs:complexContent>
        <xs:extension base="tControlBlock">
            <xs:sequence>
                <xs:element name="MinTime" type="tDurationInMilliSec" minOccurs="0"/>
                <xs:element name="MaxTime" type="tDurationInMilliSec" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes have the following meaning as shown in Table 37.

Table 37 – Attributes of the GSE element

Attribute	Description
desc	Textual description
ldInst	The instance identification of the LD within this IED, on which the control block is located. An LN is not necessary, as these control blocks are only in LLN0.
cbName	The name of the control block within the LLN0 of the LD ldInst.
cbUuid	The UUID of the control block which is referenced by this GSE. Optional.

The Address element contains the GSE address parameters in the same syntax as the server address. The appropriate P *type* values are defined in the appropriate SCSMs.

The Mintime and Maxtime elements specify the following times:

Mintime	the sending delay on a data change between the first immediate sending of the change and the first repetition in ms.
Maxtime	the source supervision time in ms (supervision heartbeat cycle time). Within this time, a failed message from the source shall be detected by the subscriber.

Mintime and Maxtime may influence SCSM parameters. Which parameters and how they are influenced is defined in the appropriate SCSM.

9.4.5 SMV address definition

The SMV element defines the address for a sampled value control block, like the GSE element does for the GSE control blocks. It is also based on the tControlBlock schema type, and therefore has the same attributes as the GSE control block.

```
<xs:complexType name="tSMV">
    <xs:complexContent>
        <xs:extension base="tControlBlock"/>
    </xs:complexContent>
</xs:complexType>
```

The attributes have the following meanings as shown in Table 38.

Table 38 – Attributes of the SMV element

Attribute	Description
desc	Textual description
ldInst	The instance identification of the LD within this IED, on which the control block is located. An LN is not necessary, as these control blocks are only in LLN0.
cbName	The name of the control block within the LLN0 of the LD ldInst.
cbUuid	The UUID of the control block which is referenced by this SMV. Optional.

The Address element contains the SMV address parameters in the same syntax as the server address. The appropriate P *type* values are defined in the appropriate SCSMs.

9.4.6 Physical connection parameters

The element PhysConn defines the type(s) of physical connection for this access point. The parameter values depend on the type of physical connection, and their types (meaning) have to be defined in the stack mapping. Additional types may be introduced for documentation purposes.

```
<xs:complexType name="tPhysConn">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="P" type="tP_PhysConn" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
```

```
            <xs:attribute name="type" type="tPhysConnTypeEnum" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tP_PhysConn">
    <xs:simpleContent>
        <xs:extension base="tPAddr">
            <xs:attribute name="type" type="tPTypePhysConnEnum" use="required"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

The type attribute specifies the type of physical connection of this access point to the bus, while the value then specifies the instance of this type (for example type="Plug", value is "ST"). The *PhysConn* type *Connection* defines a first physical connection, while the type *RedConn* can identify an additional physically redundant connection at the same access point, e.g. for redundancy protocol types PRP and HSR. Allowed types and values shall be defined in the stack mapping. The P element can be repeated with other types, if one value is not sufficient. For the physical connections defined in IEC 61850-8-1, the types and corresponding values as shown in Table 39 shall be used.

Table 39 – PhysConn P-Type definitions

PhysConn *type*	P *type*	Recommended values (IEC 61850-8-1-related)
Connection, RedConn	Type	*10BaseT, 100BaseT* etc. for electrical connection *FOC* for optical connection *Radio* for radio connection, for example WLAN
	Plug	*RJ45* for electrical plug *ST* for bajonet plug (optical glass)
	Cable	The identification of a physical cable for this connection, which connects this connection point to another connection point
	Port	The identification of a port or terminal at this access point to which a cable is connected (see connection point above) or may be connected

Restrictions

- The PhysConn *type* values as well as its P parameter *type* values shall start with a capital letter, and contain only alphanumeric characters.
- The P parameter *type* values shall be unique within each PhysConn element.
- Only one PhysConn type *RedConn* is allowed per access point, i.e. only one physically redundant connection; and if it is there, only one *PhysConn* type *Connection* is allowed, which belongs to it.

9.4.7 Communication section example

The following SCL part shows a communication section with one subnetwork W01, to which two IEDs are connected with their access points S1. The protocol type 8-MMS specifies a protocol as defined in IEC 61850-8-1 and IEC 61850-9-2. The PhysConn and address types are just examples. One IED also contains a GSE control block with an address, however without the *MaxTime* and *MinTime* elements, which are optional. Another IED contains a sampled value control block.

```
<Communication>
    <SubNetwork name="W01" type="8-MMS" uuid="176659a3-3c06-413b-b595-9d0e2a1c062e">
        <Text>Station bus</Text>
        <BitRate unit="b/s">10</BitRate>
        <ConnectedAP iedName="D1Q1SB4" apName="S1" apUuid="513d767b-410a-46c6-9afa-f9cea591768e">
            <Address>
                <P type="IP">10.0.0.11</P>
                <P type="IP-SUBNET">255.255.255.0</P>
                <P type="IP-GATEWAY">10.0.0.101</P>
                <P type="OSI-TSEL">00000001</P>
                <P type="OSI-PSEL">01</P>
                <P type="OSI-SSEL">01</P>
            </Address>
            <GSE ldInst="C1" cbName="SyckResult" cbUuid="8066999f-ff5f-4d0f-842d-977d0a43bec2">
                <Address>
                    <P type="MAC-Address">01-0C-CD-01-00-02</P>
                    <P type="APPID">3001</P>
                    <P type="VLAN-PRIORITY">4</P>
                </Address>
                <MinTime unit="s">4</MinTime>
                <MaxTime unit="s">1000</MaxTime>
            </GSE>
            <PhysConn type="Connection">
                <P type="Type">FOC</P>
                <P type="Plug">ST</P>
            </PhysConn>
        </ConnectedAP>
        <ConnectedAP iedName="E1Q1SB1" apName="S1" apUuid="64cf6be2-155e-4d74-b731-7578e74704c7">
            <Address>
                <P type="IP">10.0.0.1</P>
                <P type="IP-SUBNET">255.255.255.0</P>
                <P type="IP-GATEWAY">10.0.0.101</P>
```

```
            <P type="OSI-TSEL">00000001</P>
            <P type="OSI-PSEL">01</P>
            <P type="OSI-SSEL">01</P>
        </Address>
        <GSE ldInst="C1" cbName="ltlPositions" cbUuid="e39667ca-e306-47e2-adf6-67c8cf56be60">
            <Address>
                <P type="MAC-Address">01-0C-CD-01-00-01</P>
                <P type="APPID">3000</P>
                <P type="VLAN-PRIORITY">4</P>
            </Address>
        </GSE>
        <SMV ldInst="C1" cbName="Volt" cbUuid="695bbe00-5fef-4088-8b06-5923e499c5fd">
            <Address>
                <P type="MAC-Address">01-0C-CD-04-00-01</P>
                <P type="APPID">4000</P>
                <P type="VLAN-ID">123</P>
                <P type="VLAN-PRIORITY">4</P>
            </Address>
        </SMV>
    </ConnectedAP>
  </SubNetwork>
</Communication>
```

9.5 Data type templates

9.5.1 General

This clause defines instantiable logical node types. A logical node type is an instantiable template of the data of a logical node. The content of a logical node type shall follow the definitions in IEC 61850-7-3 and IEC 61850-7-4xx and possible vendor-specific extensions that shall follow the rules defined in IEC 61850-7-1:2011, Clause 14. A LNodeType (elsewhere also called LN type) is referenced each time that this type is or shall be instantiated within an IED. A logical node type template is built from data objects (DO) elements, which again have a DO type, which is derived from the DATA classes (CDC) defined in IEC 61850-7-3. DOs or better DOType's consist of attributes (DA) or of elements of already defined DO types (SDO). The attribute (DA) has a functional constraint, and can either have a basic type, be an enumeration, or a structure of a DAType. The DAType is built from BDA elements, defining the structure elements, which again can be BDA elements or have a basic type such as a DA. An SCT may be required to change the id of an LnodeType, DOType, DAType or EnumType but it is not allowed to change the underlying structure.

All types are uniquely identified by their id. On generation of the system SCD file from IED ICD files, the DataTypeTemplates type identifications may have to change to keep uniqueness across all IED definitions.

If a DO type is generally valid for several IEDs of different type, then the iedType attribute shall be defined as an empty string. If it is important to keep the relation of the DOType to the IED type, then iedType should be set in IED ICD to a value meaningful for a specific range of IED. Especially if an IED configurator needs the DOType contents back unchanged, it shall bind the DOType to the IED type by setting the iedType attribute.

The order of DO elements within a LNodeType definition, and of SDO/DA elements (see 9.5.3) within a DOType definition shall also specify the order of data values within a message, if this is not specified elsewhere, for example by explicit FCDA definitions in a data set down to the attribute. The order in the LNodeType definition is the responsibility of the IED configurator tool, while the order in the data set is the responsibility of the system configurator tool.

The following UML figure (Figure 23) gives an overview of the DataTypeTemplate section of the Schema.

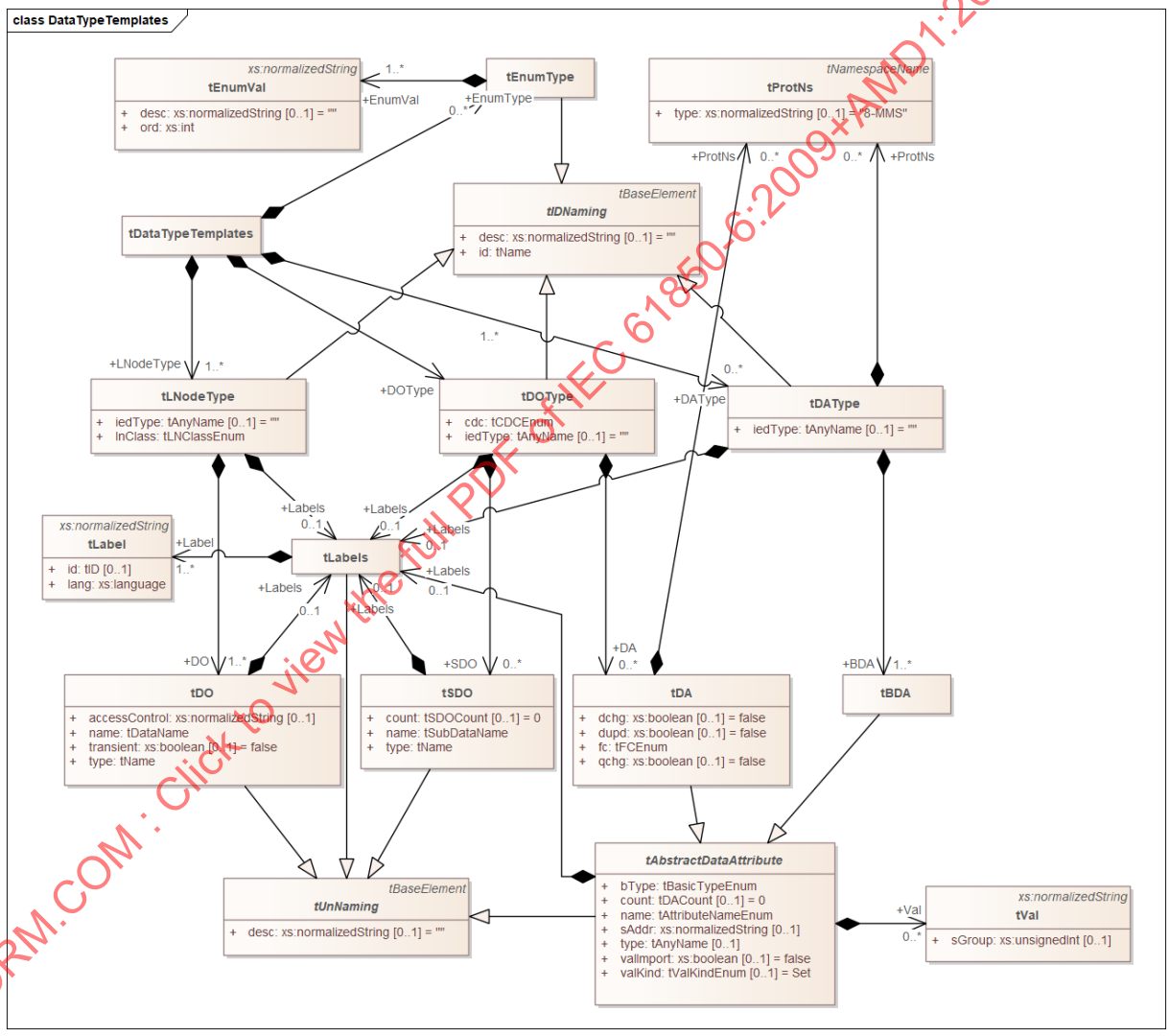


Figure 23 – UML overview of DataTypeTemplate section

All identifiers (XML attribute id) of all elements in the DataTypeTemplate section have a maximum allowed length of 255 characters and are restricted to xs:token without blanks.

The XML schema definition, inclusive defined restrictions within *DataTypeTemplates*, is as follows:

```
<xs:element name="DataTypeTemplates" type="tDataTypeTemplates">
    <xs:key name="DOTypeKey">
        <xs:selector xpath="scl:DOType"/>
        <xs:field xpath="@id"/>
    </xs:key>
    <xs:keyref name="ref2DOType" refer="DOTypeKey">
        <xs:selector xpath="scl:LNodeType/scl:DO"/>
        <xs:field xpath="@type"/>
    </xs:keyref>
    <xs:keyref name="ref2DOTypeForSDO" refer="DOTypeKey">
        <xs:selector xpath="scl:DOType/scl:SDO"/>
        <xs:field xpath="@type"/>
    </xs:keyref>
    <xs:unique name="uniqueDTT_ID">
        <xs:selector xpath="*"/>
        <xs:field xpath="@id"/>
    </xs:unique>
    <xs:key name="EnumTypeDATypeKey">
        <xs:selector xpath="scl:DAType | scl:EnumType"/>
        <xs:field xpath="@id"/>
    </xs:key>
    <xs:keyref name="ref2EnumTypeDAType" refer="scl:EnumTypeDATypeKey">
        <xs:selector xpath="scl:DOType/scl:DA | scl:DAType/scl:BDA"/>
        <xs:field xpath="@type"/>
    </xs:keyref>
</xs:element>

<xs:complexType name="tDataTypeTemplates">
    <xs:sequence>
        <xs:element name="LNodeType" type="tLNodeType" maxOccurs="unbounded">
            <xs:unique name="uniqueDOInLNodeType">
                <xs:selector xpath="scl:DO"/>
                <xs:field xpath="@name"/>
            </xs:unique>
        </xs:element>
        <xs:element name="DOType" type="tDOType" maxOccurs="unbounded">
```

```
            <xs:unique name="uniqueDAorSDOInLDOType">
                <xs:selector xpath="./*"/>
                <xs:field xpath="@name"/>
            </xs:unique>
        </xs:element>
        <xs:element name="DAType" type="tDAType" minOccurs="0" maxOccurs="unbounded">
            <xs:unique name="uniqueBDAInLDAType">
                <xs:selector xpath="scl:BDA"/>
                <xs:field xpath="@name"/>
            </xs:unique>
            <xs:unique name="uniqueProtNs">
                <xs:selector xpath="scl:ProtNs"/>
                <xs:field xpath="@type"/>
                <xs:field xpath="."/>
            </xs:unique>
        </xs:element>
        <xs:element name="EnumType" type="tEnumType" minOccurs="0" maxOccurs="unbounded">
            <xs:unique name="uniqueOrdInEnumType">
                <xs:selector xpath="scl:EnumVal"/>
                <xs:field xpath="@ord"/>
            </xs:unique>
            <xs:unique name="uniqueEnumValue">
                <xs:selector xpath="scl:EnumVal"/>
                <xs:field xpath="."/>
            </xs:unique>
        </xs:element>
    </xs:sequence>
</xs:complexType>
```

In SCL, all types are contained in the DataTypeTemplates section. As can be seen by the schema part above, the type definitions shown in Table 40 can appear there.

Table 40 – Template definition elements

Element name of Template part	Description
LNodeType	An instantiable logical node type, as referenced from IEDs and from the Substation section
DOType	An instantiable data object type; referenced from LNodeType or from the SDO element of another DOType. Instantiable version based on the CDC definitions from IEC 61850-7-3
DAType	An instantiable structured attribute type; referenced from within a DA element of a DOType, or from within another DAType for nested type definitions. Based on the attribute structure definitions of IEC 61850-7-3
EnumType	An enumeration type; referenced from the DA element of a DOType or from a DAType, in case that the *bType* is Enum. The definitions shall follow enumeration definitions from IEC 61850-7-3 and IEC 61850-7-4 as well as other domain standards.

9.5.2 LNodeType definitions

The LN type (LNodeType element) contains a list of data objects (DO), its attributes, and possible default values for configuration parameters.

```
<xs:complexType name="tLNodeType">
    <xs:complexContent>
        <xs:extension base="tIDNaming">
            <xs:sequence>
                <xs:element name="DO" type="tDO" maxOccurs="unbounded"/>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInLNode">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="iedType" type="tAnyName" use="optional" default=""/>
            <xs:attribute name="lnClass" type="tLNClassEnum" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes have the following meaning as shown in Table 41.

Table 41 – Attributes of the LNodeType element

Attribute	Description
id	A reference identifying this LN type within this SCL section; used by the LN attribute LNType to reference this definition
desc	An additional text describing this LN type
iedType	The manufacturer IED type of the IED to which this LN type belongs – deprecated to keep LNodeTypes generic. The IED constraints needs to be handled at DOType level, and not allowed at LNodeType level from edition 2.2 and onward.
lnClass	The LN base class of this type as specified in IEC 61850-7-x; observe that here an enumeration exists, which allows extensions (names containing only capital letters)

The DO element references the instantiable data type of this DO.

```
<xs:complexType name="tDO">
    <xs:annotation>
        <xs:documentation xml:lang="en">See Section 9.5.1</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInLNode">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="tDataName" use="required"/>
            <xs:attribute name="type" type="tName" use="required"/>
            <xs:attribute name="accessControl" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="transient" type="xs:boolean" use="optional" default="false"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The DO attributes are used as shown in Table 42.

Table 42 – Attributes of the DO element

Attribute	Description
name	The data object name as specified for example in IEC 61850-7-4
type	The *type* references the *id* of a DOType definition
accessControl	Access control definition for this DO. If it is missing, then any higher-level access control definition applies
transient	If set to true, it indicates that the Transient definition from IEC 61850-7-4 applies
desc	Descriptive text for the DO element

9.5.3 DO type definition

The DOType element referenced by the *type* attribute of the LNodeType DO element has the following syntax:

```
<xs:complexType name="tDOType">
    <xs:complexContent>
        <xs:extension base="tIDNaming">
            <xs:sequence>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="SDO" type="tSDO"/>
                    <xs:element name="DA" type="tDA">
                        <xs:unique name="uniqueProtNsInDA">
                            <xs:selector xpath="scl:ProtNs"/>
                            <xs:field xpath="@type"/>
                            <xs:field xpath="."/>
                        </xs:unique>
                    </xs:element>
                </xs:choice>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInLNode">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="iedType" type="tAnyName" use="optional" default=""/>
            <xs:attribute name="cdc" type="tCDCEnum" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The DOType identifies the contents of the DO. This can be either attributes (DA elements), or the reference to another DOType (SDO element). Although in general the DOType definition might be empty e.g. for an SSD or an SED file, it shall contain at least one SDO or DA element for SCD and ICD files.

The attributes have the following meaning as shown in Table 43.

Table 43 – Attributes of the DOType element

Attribute	Description
id	The (SCL file global) identification of this DOType. Used to reference this type.
iedType	The type of the IED to which this DOType belongs. The empty string respective a missing attribute allows references for all IED types, or from the Substation section without IED identification.
cdc	The basic CDC (Common Data Class) as defined in IEC 61850-7-3.
desc	Description of this DOType

Restrictions

- IEC 61850-7-1:2011/AMD1:2020 has improved the namespace extension to allow Transitional namespaces to extend the list of CDCs for future works. This extension is allowed by SCL only in this specific context and shall not be done in any other circumstances.

The SDO element then references another DOType definition.

Warning: recursive references are not allowed, but cannot be checked at syntax level!

```
<xs:simpleType name="tSDOCount">
    <xs:union memberTypes="xs:unsignedInt tRestrName1stL"/>
</xs:simpleType>

<xs:complexType name="tSDO">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInLNode">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="tSubDataName" use="required"/>
            <xs:attribute name="type" type="tName" use="required"/>
            <xs:attribute name="count" type="scl:tSDOCount" use="optional" default="0"/>
```

```
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes of the SDO element are defined in Table 44.

Table 44 – Attributes of the SDO element

Attribute	Description
name	The SDO name
desc	Descriptive text for the SDO
type	References the DOType defining the contents of the SDO
count	The number or reference to an attribute defining the number of array elements, if this element has an ARRAY type. If missing, the default value is 0 (no array). The usage of number is kept only for backward compatibility.

The attribute (DA) definition carries the handling attributes according to IEC 61850-7-3 as defined in the appropriate tables. Each instantiable attribute shall be defined in the DO type definition. Observe that a certain SCSM (for example IEC 61850-8-1) might define additional mandatory attributes or SDOs. The DA syntax is described in 9.5.4.

9.5.4 Data attribute (DA) definition

9.5.4.1 General

The DA element defines the attributes, their stack-related handling, and describes their (default) values or specifies typical values for all instances.

The DA element has either a basic type, or again a reference to a structured attribute type definition for example in the case of an attribute with a structure such as *ScaledValueConfig*. If the DA is an array, then its count attribute gives the number of array elements or, respectively, references the attribute which contains it. IEC 61850-7-3 and for some enumerations IEC 61850-7-4 define the type of a certain attribute based on the CDC of the DO.

The value coding syntax in the Val element of the DA element then has to follow the XML schema data type coding definitions for the IEC 61850-7 basic data types. The type mapping is as shown in Table 45 as such as the corresponding attribute type in SCL (DA bType).

Table 45 – Data type mapping

IEC 61850-7-2 basic type	XML Schema (xs) data type	SCL type (attribute bType)	Value representation
INT8, INT16, INT32, INT64 INT8U, INT16U, INT24U, INT32U	integer	INT8, INT16, INT32, INT64 INT8U, INT16U, INT24U, INT32U	An integer number, no decimal fraction (99999)
FLOAT32	double	FLOAT32	A number with or without a decimal fraction (+999.99999), or with an exponent (+9.999999e+999)
BOOLEAN	boolean	BOOLEAN	*false*, *true* or 0, 1
ENUMERATED	normalizedString	Enum	The enumeration element names as defined in the EnumType associated with the DA element as string values

IEC 61850-7-2 basic type	XML Schema (xs) data type	SCL type (attribute bType)	Value representation
Octet64	base64Binary	Octet64	Coding according to 6.8 of RFC 2045; observe that SCSMs might define another representation for certain attributes or properties
VisString64, VisString129, VisString255	normalizedString	VisString64, VisString129, VisString255	A character string without tabs, linefeeds and carriage return, restricted to 8-bit characters (ISO/IEC 8859-1 characters limited to UTF-8 single byte coding)
Unicode255	normalizedString	Unicode255	A character string without tabs, linefeeds and carriage return. All characters in an XML file are principally Unicode, for example in UTF-8 coding
PhyComAddr	-	PhyComAddr	The PhyComAddr value is not intended to be specified in SCL.
ObjectReference	normalizedString	ObjRef	The reference to an IEC 61850 object, as defined in IEC 61850-7-2. Additionally IED internal references especially in ICD files shall be stated as follows: @ldInst/lnName[.doName[.attributename]], i.e. the @ replaces the IED name. An ObjectReference may also use UUID in the SCL representation, by indicating #UID followed by the UUID itself, replacing the referenced element and all its hierarchy, as follows: #UIDxxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx[.doName[.attributename]]. The #UID could replace LDevice, LN or Control Block. Observe that the appropriate online values always must be absolute names considering any ldName or UUID values, i.e. the relative names as such as the UUID are only allowed inside SCL.
EntryID	-	EntryID	The EntryID value is not intended to be specified in SCL.
Timestamp (UTC time)	dateTime	Timestamp	Coding without time zone, e.g. 2007-12-31T21:01:12.345
Currency	normalizedString	Currency	See IEC 61850-7-2: values are coded according to ISO 4217 3-character currency code
Quality		Quality	The Quality value is not intended to be specified in SCL.
EntryTime	-	EntryTime	The EntryTime value is not intended to be specified in SCL.
TriggerConditions	-	TrgOps	The TriggerConditions are specified by mean of specific structure in corresponding control block.
RCBReportOptions	-	OptFlds	The RCBReportOptions are specified by mean of specific structure in a report control block.
LCBLogEntryOptions	-	LogOptFlds	The LCBLogEntryOptions are specified by mean of specific structure in a log control block.
SVMessageOptions	-	SvOptFlds	The SVMessageOptions are specified by mean of specific structure in a sampled value control block.
CheckConditions	-	Checks	The Checks value is not intended to be specified in SCL.
Step control (Coded Enum)	-	Tcmd	The Step control value is not intended to be specified in SCL.

IEC 61850-7-2 basic type	XML Schema (xs) data type	SCL type (attribute bType)	Value representation
Double point status (Coded Enum)	-	DbPos	The Double point status value is not intended to be specified in SCL.

The management of value for ObjectReference type needs to consider different levels. The value can be set by an ICT. If the value needs to be set or updated by the SCT when names of the referred object are updated (typically IED name, *ldName*, LN prefix and instance and control block name), then it is recommended to have *valImport* of such DA set to true to allow the SCT to update the value. This is the responsibility of the ICT to set *valImport* correctly or to ensure coherency of the value if SCT is not allowed to update it. When *valImport* is true, the SCT shall maintain the ObjectReference on updates of the referred object.

The meaning of the value for an IED configurator can be different depending on the device capabilities, the functional characteristic of the attribute, and the stage of the engineering process. The DA attribute *valKind* allows the specification of this meaning. It is ignored for all cases not specified in Table 46 (for example for the *q* and *t* attributes). The attributes *valKind* and *valImport* make no sense for DA elements with bType = "Struct" and shall then be ignored by the tools.

```
<xs:simpleType name="tValKindEnum">
    <xs:restriction base="xs:Name">
        <xs:enumeration value="Spec"/>
        <xs:enumeration value="Conf"/>
        <xs:enumeration value="RO"/>
        <xs:enumeration value="Set"/>
    </xs:restriction>
</xs:simpleType>
```

Table 46 – Attribute value kind (valKind) meaning

valKind value	Functional constraints	Engineering process stage	Meaning
Spec	Non operational (CF, DC)	Specification phase	The wanted value determined at specification phase typically in an SSD file
Conf	CF, DC, operational attribute of a CDC used for settings, EX (attribute dataNs only)	IED template, after IED engineering	This value is not visible online at the IED. The IED is engineered such that this value is used
RO	Operational process state attribute	IED template	The default value for the attribute to be used if the value is fix on the IED
RO	CF, DC, operational attribute of data used for settings	IED template, after IED configuration	Read only value at an IED – can only be set at configuration time
Set	CF, DC	At/after IED configuration	A determined setting value. The value is/shall be set within the IED
Set	Operational process values (except time and quality)	At/after IED configuration (possibly RO changed to Set)	The default value for the operational attribute, e.g. for startup or simulation
Set	Operational setting value (SP, SG for all data used as setting)	At/after IED configuration	The setting value for the set point respectively parameter

This allows, for example, the definition of IED capabilities (which attributes are available, which are read only), the default values an IED is delivered with (readable, changeable, or not visible at all), or the setting values for operative (for example protection) parameters.

The additional attribute *valImport* allows to define if an IED / IED configurator can import values modified by another tool from an SCD file, even if valKind=RO or valKind=Conf. The attribute *valImport* makes no sense for DA elements with bType = "Struct" and shall then be ignored by the tools.

The syntax definition is as follows. It is based on an abstract type tAbstractDataAttribute which is reused later in attribute structure definitions.

```
<xs:complexType name="tDA">
    <xs:complexContent>
        <xs:extension base="tAbstractDataAttribute">
            <xs:sequence>
                <xs:element name="ProtNs" type="tProtNs" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attributeGroup ref="agDATrgOp"/>
            <xs:attribute name="fc" type="tFCEnum" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>

<xs:attributeGroup name="agDATrgOp">
    <xs:attribute name="dchg" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="qchg" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="dupd" type="xs:boolean" use="optional" default="false"/>
</xs:attributeGroup>

<xs:complexType name="tAbstractDataAttribute" abstract="true">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Val" type="tVal" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
                    <xs:unique name="uniqueLabelInLNode">
                        <xs:selector xpath="./scl:Label"/>
                        <xs:field xpath="@id"/>
                        <xs:field xpath="@lang"/>
                    </xs:unique>
                </xs:element>
```

```
            </xs:sequence>
            <xs:attribute name="name" type="tAttributeNameEnum" use="required"/>
            <xs:attribute name="sAddr" use="optional">
                <xs:simpleType>
                    <xs:restriction base="xs:normalizedString">
                        <xs:maxLength value="255"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="bType" type="tBasicTypeEnum" use="required"/>
            <xs:attribute name="valKind" type="tValKindEnum" use="optional" default="Set"/>
            <xs:attribute name="type" type="tAnyName" use="optional"/>
            <xs:attribute name="count" type="tDACount" use="optional" default="0"/>
            <xs:attribute name="valImport" type="xs:boolean" use="optional" default="false"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The attributes of the DA element are defined in Table 47.

Table 47 – Attributes of the DA element

Attribute	Description
desc	Some descriptive text for the attribute
name	The attribute name; the type tAttributeNameEnum restricts to the attribute names from IEC 61850-7-3, plus new ones starting with lower case letters
fc	The functional constraint for this attribute; *fc*=SE always also implies *fc*=SG; *fc*=SG means that the values are visible, but not editable
dchg, qchg, dupd	Defines which trigger option is supported by the attribute (value true means supported). One of those allowed according to IEC61850-7-3 shall be chosen.
sAddr	An optional short address of this attribute (see 9.5.4.3)
bType	The basic type of the attribute, taken from tBasicTypeEnum (see 9.5.4.2)
type	Shall only be used if *bType*= Enum or *bType* = Struct to refer to the appropriate enumeration type or DAType (attribute structure) definition
count	Optional. Shall state the number of array elements or reference the attribute stating this number in case that this attribute is an array. A referenced attribute shall exist in the same type definition. The default value 0 states that the attribute is no array. The usage of number is kept only for backward compatibility.
valKind	Determines how the value shall be interpreted if any is given – see Table 46
valImport	if true, an IED or IED configurator can import values modified by another tool from an SCD file, even if valKind=RO or valKind=Conf. It is the responsibility of the IED configurator to assure value consistency and value allowance even if valImport is true.

The attributes *name, fc*, and *bType* shall always be defined. All instantiable attributes contained within a DO shall be defined.

In case that the DA element has a basic type and belongs to a specific stack mapping, it shall contain the ProtNS element, which defines the stack mapping(s) to which it belongs (see 9.5.5).

9.5.4.2 Attribute basic types

The basic types allowed are as follows:

```
<xs:simpleType name="tPredefinedBasicTypeEnum">
    <xs:restriction base="xs:Name">
        <xs:enumeration value="BOOLEAN"/>
        <xs:enumeration value="INT8"/>
        <xs:enumeration value="INT16"/>
        <xs:enumeration value="INT24"/>
        <xs:enumeration value="INT32"/>
        <xs:enumeration value="INT64"/>
        <xs:enumeration value="INT128"/>
        <xs:enumeration value="INT8U"/>
        <xs:enumeration value="INT16U"/>
        <xs:enumeration value="INT24U"/>
        <xs:enumeration value="INT32U"/>
        <xs:enumeration value="FLOAT32"/>
        <xs:enumeration value="FLOAT64"/>
        <xs:enumeration value="Enum"/>
        <xs:enumeration value="Dbpos"/>
        <xs:enumeration value="Tcmd"/>
        <xs:enumeration value="Quality"/>
        <xs:enumeration value="Timestamp"/>
        <xs:enumeration value="VisString32"/>
        <xs:enumeration value="VisString64"/>
        <xs:enumeration value="VisString65"/>
        <xs:enumeration value="VisString129"/>
        <xs:enumeration value="VisString255"/>
        <xs:enumeration value="Octet64"/>
        <xs:enumeration value="Unicode255"/>
        <xs:enumeration value="Struct"/>
        <xs:enumeration value="EntryTime"/>
        <xs:enumeration value="Check"/>
        <xs:enumeration value="ObjRef"/>
        <xs:enumeration value="Currency"/>
        <xs:enumeration value="PhyComAddr"/>
```

```
            <xs:enumeration value="TrgOps"/>
            <xs:enumeration value="OptFlds"/>
            <xs:enumeration value="SvOptFlds"/>
            <xs:enumeration value="LogOptFlds"/>
            <xs:enumeration value="EntryID"/>
            <!-- for 61850-8-1 Edition 2.1 -->
            <xs:enumeration value="Octet6"/>
            <xs:enumeration value="Octet16"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tBasicTypeEnum">
        <xs:restriction base="tPredefinedBasicTypeEnum"/>
    </xs:simpleType>
```

tPredefinedBasicTypeEnum contains the definitions as defined in IEC 61850-7-x. CODED ENUMs are replaced by concrete basic types Quality, Dbpos for double bit positions as used in DPC and DPS, and Tcmd for tap changer commands used in BSC. Check is introduced for the appropriate data attribute used in IEC 61850-8-1. Quality, Check, Dbpos and Tcmd remain opaque (no values required in SCL). Similarly PhyComAddr, SvOptFlds, OptFlds, LogOptFlds and TrgOps remain opaque, just for usage in the common data classes for service tracking. For VisibleString, UnicodeString and OctetString length dependent (sub-)types are introduced. VisString32 is for example a VisibleString of maximum length of 32 characters. *ObjRef* is basically a string type, which contains the reference to another IEC 61850 object as defined in IEC 61850-7-2, where the maximum allowed length is also specified.

NOTE 1 INT128 and INT24U exist only for backwards compatibility reasons, and are deprecated.

NOTE 2 In contrast to the 2003 version of this standard *tPredefinedBasicTypeEnum* no longer allows extensions of the base types.

tPredefinedBasicTypeEnum will be used for the schema of this version. It should be kept in mind when developing tools that, e.g. after extensions in other standards, also other types according to *tBasicTypeEnum* should be syntactically accepted as input – and should be handled e.g. with *mayIgnore* or *mustUnderstand* rules.

The following example defines the stVal attribute of a DPC CDC without value, according to IEC 61850-7-3:

```
    <DA name="stVal" fc="ST" dchg="true" bType="Dbpos"/>
```

9.5.4.3 Short addresses

The *sAddr* attribute allows the allocation of a short address to DO attributes. Short addresses can be used within the communication to make them more efficient either in the communication, or in the handling of messages at client or server. Furthermore, they can be used as IED internal identification for the attribute. To be able to use short addresses in the communication,

- the stack mapping must allow them and define their meaning, or
- the IED must allow them.

The detailed syntax of a short address value depends on the stack if the stack (SCSM) defines their usage, or else on the IED tool. SCL foresees a two level hierarchy for short addresses used in communication:

1) the communication address of the IED/server/access point;
2) the short address of a data item at attribute level.

It is possible to use the short address instead of the (symbolic) IED communication address if the short address is unique system-wide, and the stack (SCSM) allows this. Otherwise, the short address value scope and syntax is private to the IED.

Tools which do not handle short addresses shall also preserve imported contents in exported SCL files.

9.5.4.4 Values

The optional value definition contains one value. The XML coding of the value is defined in 9.5.4.1 respective Table 45. For attributes with *fc* = **SG**, the *sGroup* attribute specifies to which setting group this value belongs. There may be a value for each defined setting group. The meaning of the value in the engineering process is defined at the DA/DAI level by means of the *valKind* attribute.

```
<xs:complexType name="tVal">
    <xs:simpleContent>
        <xs:extension base="xs:normalizedString">
            <xs:attribute name="sGroup" type="xs:unsignedInt" use="optional"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
```

Attribute description

sGroup the number of the setting group (if *fc* = "SG") to which this value belongs.

The *sGroup* value used within an IED should be checked against an existing setting group definition on this IED / LD, where the maximum allowed number is specified (SettingControl.numOfSGs). If the optional *sGroup* attribute is missing completely, then either the concerned DATA attribute is in no setting group (*fc* # SG), or the data value applies to all setting groups.

9.5.5 Data attribute structure type

If the DA.*bType* value is Struct, the DA.*type* attribute references an attribute structure. These structures are defined with DAType elements.

```
<xs:complexType name="tDAType">
    <xs:complexContent>
        <xs:extension base="tIDNaming">
            <xs:sequence>
                <xs:element name="BDA" type="tBDA" maxOccurs="unbounded"/>
                <xs:element name="ProtNs" type="tProtNs" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="Labels" type="tLabels" minOccurs="0">
```

```
                <xs:unique name="uniqueLabelInLNode">
                    <xs:selector xpath="./scl:Label"/>
                    <xs:field xpath="@id"/>
                    <xs:field xpath="@lang"/>
                </xs:unique>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="iedType" type="tAnyName" use="optional" default=""/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="tProtNs">
  <xs:simpleContent>
    <xs:extension base="tNamespaceName">
      <xs:attribute name="type" use="optional" default="8-MMS">
        <xs:simpleType>
          <xs:restriction base="xs:normalizedString">
            <xs:minLength value="1"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

The DAType element contains a list of attributes with the BDA element. These attributes can either have a basic type or refer to another attribute structure. The definitions have to follow IEC 61850-7-3 in structure, type and naming.

```
<xs:complexType name="tBDA">
  <xs:complexContent>
    <xs:extension base="tAbstractDataAttribute"/>
  </xs:complexContent>
</xs:complexType>
```

The BDA element instantiates the tAbstractDataAttribute and has therefore the same attributes.

The attributes of the BDA element are defined in Table 48.

Table 48 – Attributes of the BDA element

Attribute	Description
desc	Some descriptive text for the attribute
name	The attribute name; the type tAttributeEnum restricts to the attribute names from IEC 61850-7-3, plus new ones starting with lower case letters
sAddr	an optional short address of this BDA attribute
bType	The basic type of the attribute, taken from tBasicTypeEnum
type	Only used if *bType*= Enum or *bType* = Struct to refer to the appropriate enumeration type or DAType definition
count	Optional. Shall state the number of array elements in the case where the attribute is an array
valKind	Determines how the value shall be interpreted if any is given – see Table 46

Note that the sAddr attribute might appear on several levels, starting with the DA element. There are in principle two methods to handle this:

1) use only the lowest level value;
2) use values on all levels as a kind of hierarchical short address.

It is up to the SCSM, respectively the IED tool, to decide which method is used (see also 9.5.4.3).

For *valKind* only the lowest level value shall be used.

If the DA type definition belongs to a specific stack mapping like the Oper structure for the Operate service as defined in IEC 61850-8-1, then the BDA list shall be followed by a ProtNS element for each SCSM, which needs this specific DA type. The ProtNS element has a type attribute defining the protocol with default value 8-MMS for the mapping defined in IEC 61850-8-1, and its contents defines the version of this mapping. An example ProtNs element for a mapping according to the IEC 61850-8-1 version from 2003 follows here:

```
<ProtNs type="8-MMS">IEC 61850-8-1:2003</ProtNs>
```

9.5.6 Enumeration types

Enumerations are in general used in more than one LNodeType. Therefore, an enumeration type definition is made for them.

```
<xs:complexType name="tEnumType">
    <xs:complexContent>
        <xs:extension base="tIDNaming">
            <xs:sequence>
                <xs:element name="EnumVal" type="tEnumVal" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

Enumeration definitions within a SCD file are valid for all IEDs; they are not IED type-dependent. Enumeration definitions are given in parts 7-2, 7-3 and 7-4 and other domain standards. Observe that these definitions may be enhanced by new versions of these standards. If private extensions of these enumerations are used as allowed by these standards, or private enumerations are defined, the name must indicate this appropriately, i.e. none of the standard defined names should be used. This is especially important for extensions, because different manufacturers might use different extensions. It is also important, if only a subrange of the enumeration value set is supported. The supported subset shall be indicated within an ICD file by an IED private enumeration type, where the unsupported values are missing.

If the semantics of the same LN class code and same data object name code for an enumeration in another IEC name space is redefined, then the enumeration type and its values shall also be kept unchanged (possibly with redefined semantics or with value extensions).

The meaning of the attributes of the EnumType element is as shown in Table 49.

Table 49 – Attributes of the EnumType element

Attribute	Description
id	A name identifying this enumeration type; used by the *type* attribute of DA and BDA elements to reference this definition in the case where the *bType* is Enum. The length restriction is indicated in XSD code component.
desc	An additional text describing this Enum type

The values of the enumeration are defined as follows:

```
<xs:complexType name="tEnumVal">
    <xs:simpleContent>
        <xs:extension base="tEnumStringValue">
            <xs:attribute name="ord" type="xs:int" use="required"/>
            <xs:attributeGroup ref="agDesc"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>

<xs:simpleType name="tEnumStringValue">
    <xs:restriction base="xs:normalizedString">
        <xs:maxLength value="127"/>
        <xs:pattern value="[\p{IsBasicLatin}\p{IsLatin-1Supplement}]*"/>
    </xs:restriction>
</xs:simpleType>
```

The *ord* attribute contains the order of the values, with some exceptions explicitly defined in IEC 61850-7-3 or in IEC 61850-7-4. The value of type tEnumStringValue is the character string as defined in IEC 61850-7-3 or IEC 61850-7-4. Any private definitions also shall be restricted to 127 characters maximum length and a character set of Basic Latin or Latin-1 supplement. The *desc* attribute allows descriptive text for the meaning of the value.

9.5.7 Data type template examples

Examples can be found in the *DataTypeTemplate* section of Clause D.2.

10 Tool and project engineering rights

This clause refers to Clause 5 concerning the engineering process, the definition of roles of a system configurator and an IED configurator within a system project, and additionally a communication interface-related data exchange at system level between different projects. This clause defines the intended engineering responsibility areas in terms of the previously defined SCL elements.

10.1 IED configurator

The task of the IED configurator is to create the ICD file, and to modify the data model, parameter and configuration values either for a new ICD file or a project specific IED instance by means of an IID file. Both may contain preconfigured data sets and control blocks, and default addresses for an IED of this type. For an IID file produced from an SCD file, the already configured / used data sets and control blocks shall remain unchanged against the SCD file. Finally the IED configurator is responsible for binding incoming data from other IEDs as defined within an imported SCD file to internal signals, e.g. by means of the SCL Input section, and for generating and loading the IED instance specific configuration data, which a CID file could be a part of.

How an ICD file is created depends on the IED capabilities and the tool design. There is a big range available from a fix ICD file for use in each project (the only possible adaptations are the IED name or LD name and IED address), up to ICD file generation for a specific usage of the IED e.g. for a project specific bay type after extensive preengineering by means of the IED tool – see also Clause 7 – or a preengineered project specific IED based on the current state of the system (IID file generated based on a SCD file). An ICD file shall indicate the capabilities of a possibly preengineered IED. Therefore any enumeration values which are not supported by the IED shall be removed from the referenced enumeration types, and vendor or even IED type specific enumeration type names used. The full standardized enumeration definition for configuration data and settings shall only be used

- for the enumerations of SI units and multiplier
- if all enumeration values are supported
- if the data attribute supports only a single fix value (indicated by valKind="RO" & valImport="false"), and the value is supplied in the SCL file.

Any change in data model, parameter and configuration values shall be reflected in appropriate version indicators within the LN0 NamPlt data object as values within the SCL file. For predefined data sets and control blocks the version information has to be managed as defined in IEC 61850-7-2 for the confRev parameter of the control blocks, and also contained in the SCL file.

If an SCD file is imported, an IED tool may update the version and related value information and change parameter and configuration values as well as binding external data to internal signals. It may add new control blocks and data sets. It may modify preconfigured data sets (i.e. those originally created by it) if these are not used by other IEDs in some configured data flow. It is not allowed to modify data sets which are used by some other IED. The result is transferred back to the system configurator by means of an IID file. Only the following data model changes are allowed (but not mandatory):

- addition of logical devices, logical nodes or DATA within logical nodes;
- removal of logical devices, logical nodes or DATA, which are not referenced by some client / subscriber or bound to the primary system description (substation section).

Observe that both kinds of changes lead to new data model version identifications, which have to be reimported by the system tool and might influence the reloading of other IEDs in the system.

NOTE IED online change of data flow is described and defined in IEC 61850-7-2. Online changes of data models on an IED outside SCL are outside the scope of this standard. To keep system consistency for those IEDs already integrated into a system, they could either be prevented or follow the above offline engineering rules, especially what concerns the provision of new data model revision information.

10.2 System configurator

The tasks of the system configurator are to create IED instances from IED templates, engineer the data flow between the IEDs, give addresses to them and bind the logical nodes to the primary process. Therefore as well as instantiating IED templates, the system tool handles the following SCL sections:

- Process / Line / Substation section, including references to logical nodes on IEDs (9.2).
- Communication section including project specific instance addresses (9.4).
- Data sets and control blocks, as allowed by the IED capabilities (9.3.7 and following).
- Allocation of data flow and report control block instances to clients, as allowed by the IED capabilities (ClientLN element at report control blocks, IEDName at other control blocks).
- Creating IED input sections as seen from system engineering point of view, however without binding to IED internal signals (9.3.13).
- Binding of input sections to IED supplied input section templates.
- Reorganizing the DataTypeTemplate section (9.5) to keep the type identifiers unique and the template section short, however on condition, that the instance information is unchanged when the templates are expanded at IEDs to an instance. This concerns not only the structures and type, but also the values and Private sections defined within the types.

The system configurator shall increment the control block *confRev* values of all statically defined data sets on creation and modification as defined in IEC 61850-7-2. Further it defines the project identity by means of the SCD Header identification, and manages the SCD header revision history.

Where a system configurator also changes configuration values and parameter values for an IED, it has also to increment the appropriate paramRev and valRev attributes in the LN0 NamPlt as well as other LNs having appropriate NamPlt attributes. It is the responsibility of the system engineer to clarify before such changes are made, whether the concerned IED supports loading of this data via an SCD file.

10.3 Right transfer between projects

The right of data flow engineering can be formally transferred from one project to another project by means of a SED file. The concerned IEDs are marked with the *engRight* attribute value *dataflow*. To not lose already predefined references on these IEDs, all referenced IEDs have also to be exported at least partly, i.e. just the LDs and data sets referenced, with *engRight=fix*. If address coordination is an issue, also the access points of all IEDs with defined addresses can be exported.

Observe that also the appropriate part of the Process / Substation section and Communication section shall be exported, however only as far as they contain references to the exported IEDs, or shall be used by the other project. The primary part shall contain topologically and equipment wise complete bays. The relation between exported IED's logical nodes and the primary equipment is fix for the receiving project, however it might add references to its own IEDs into the Substation section, and shall add addresses of its own IEDs into the Communication section.

The importing project can use all received information as needed (e.g. use one of the fix IEDs as client / subscriber for its own IEDs, or add its own IEDs as clients / subscribers to provided free control block instances). It has the following additional engineering rights (modification rights) only on those IEDs exported with *dataflow* right:

- addition of data sets and control blocks as allowed by the IED capabilities. Observe that the IED owner is allowed to restrict these capabilities in the SED file;
- modification of data sets allocated purely to clients in the importing project
- adding of data set members to data sets allocated to clients in both projects.
- addition of data expected to be received from its own IEDs to the Input sections of imported IEDs.
- Binding of data sent from its own IEDs to input signal templates in imported IEDs.

When ready with engineering, the receiving project shall export again a SED file, containing all imported IEDs plus those of its own project referenced by them after engineering either with fix, or again with dataflow right. The exported SED file shall have the same Header identification as the imported SED file, however with increased revision index. An import of this file gives the originally exporting project the full engineering right back. Observe that if some IED with dataflow right of the other project is imported, that then again after engineering finalization a SED file has to be exported back. It is up to the exporting project to decide if its own IEDs are again exported with dataflow engineering right to have another engineering round, or just as fix to complete the data exchange.

Observe that IEDs exported as fix are still under full control of the exporting project and might be changed by it. If this happens, this is discovered online, at the latest, by discrepancies between assumed and actual control block and data model revision information. However, it is good engineering practice to notify the concerned project if this is noticed, e.g. by sending another SED file.

IEDs exported as dataflow have to be set to fix in their own project, which should block any changes. It is good practice to export this state along with SCD files and other SED files to other projects. This prohibits two different projects from adding data flow definitions to the same IED at the same time.

Table 52 gives an overview about allowed actions for IEDs exported with fix or data flow engineering rights.

Table 52 – Allowed SCT engineering actions

Exported right:	Fix		Dataflow	
Actions on imported/exported IED	**Exporting project (right stays full)**	**Importing project**	**Exporting project (right becomes Fix)**	**Importing project**
Add clients / subscribers to existing (free) control blocks	X (conflict possible for reporting)	X	Not allowed for exported RCBs!	X
Use as destination IED for own IEDs as senders	X	X	X	X
Add data sets and control blocks as allowed by IED capabilities	X			X
Modify data sets (delete / rearrange FCDA)	X (conflict possible)			only if all clients in own project
Add FCDA to existing data sets	X (conflict possible)			X
Add data to be received from own IEDs to the Input sections of exported IEDs.	X			X
Bind data sent from own IEDs to input signal templates of exported IEDs	X			X

Annex A
(normative)

SCL syntax: XML schema definition

A.7 General

This Annex A defines the XML schema definition files for the SCL language. Observe that in case that faults need to be fixed new schema files will be provided by IEC as separate code component.

The SCL schema definition is available as a code component at the TC57 website. For more information about the SCL code component see Subclause 1.3.

A.1 Base types

The following XML schema files define the basic types used by the whole SCL schemas:

File **SCL_BaseSimpleTypes.xsd**

File **SCL_Enums.xsd**

File **SCL_BaseTypes.xsd**

A.2 Substation syntax

This XML schema file defines the process section of the SCL.

File **SCL_Substation.xsd**

A.3 Data type templates

This XML schema file defines the data types section of the SCL.

File **SCL_DataTypeTemplates.xsd**

A.4 IED capabilities and structure

This XML schema file defines the IED section of the SCL.

File **SCL_IED.xsd**

A.5 Communication subnetworks

This XML schema file defines the communication section of the SCL.

File **SCL_Communication.xsd**

A.6 Main SCL

This XML schema file defines the main section of the SCL containing all other sections.

File **SCL.xsd**

Annex B
(informative)

SCL enumerations according to IEC 61850-7-3 and IEC 61850-7-4

The enumeration definitions related to IEC 61850-7-2, -7-3 and -7-4 can be found in the respective second editions.

Annex C
(informative)

Syntax extension examples – Extension syntax for drawing layout coordinates

This annex defines a simple SCL extension to add coordinates to objects, so that they can be easily shown on a drawing. This is sufficient for a lot of drawing tasks, and serves here as an example of an extension of the SCL language by another name space.

The handling (for example drawing) of object connections as well as the packaging of objects into drawing pages is private to the interpreting application. Typical drawings could be that of a substation as substation single line, a bay as bay single line and the communication section as a communication configuration drawing.

The coordinate system is a relative *x, y* system with coordinates using positive integer numbers. The point (0,0) is the upper left point of a drawing plane which is unlimited to downwards and right direction. The unit 1 principally refers to the size of an object. If different object sizes are used, then 1 is the size of the smallest object. However, transport of coordinates between different drawing applications might in this case lead to strange representations.

If coordinates are defined at different SCL tag hierarchy levels, then each level contains coordinates relative to the higher level. The absolute coordinate of a lower level is thus calculated by summing up all higher level coordinates and the object coordinates themselves. If there are no coordinates defined at a higher level, then (x,y) = (0,0) is assumed.

This is illustrated in Figure C.1. Here, for example, the bay 3 of Substation 1 voltageLevel 1 has the absolute coordinates (0+1+8, 0+1+4) = (9,5) within a picture showing the substation 1, or even both substations.

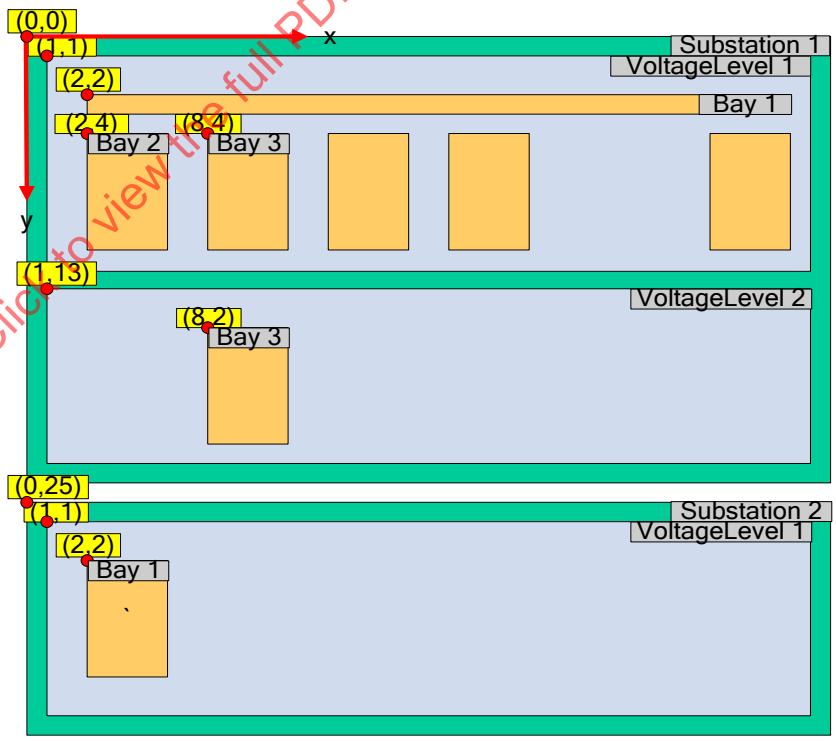


Figure C.1 – Coordinate example

Similarly the coordinates at IEDs and the SubNetworks of the Communication section can be used to place them into a communication configuration diagram. In this case, as no hierarchy is implied, the coordinates are absolute in the x,y plane.

Additional XML elements:

Only the additional XML attributes *x* and *y* for the coordinates in the x and y direction are needed in addition to the SCL elements, which represent drawable objects. Additionally, the optional attribute *dir* with the value *horizontal* or *vertical* can give the preferred connection direction of the object. If this attribute is defined at a bay, this means that all contained primary devices are oriented vertically, except those where another value of *dir* is explicitly stated. The coordinate name space shall be
http://www.iec.ch/61850/2003/SCLcoordinates.

An appropriate XML schema definition e.g. as a file SCL_Coordinates.xsd is:

```
<xs:schema targetNamespace="http://www.iec.ch/61850/2003/SCLcoordinates"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.iec.ch/61850/2003/SCLcoordinates"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1.0">
    <xs:annotation>
        <xs:documentation xml:lang="en">
                COPYRIGHT IEC, 2005. Version 1.0. Release 2005/09/11.
                This schema is for infomational purposes only, and is not normative!
        </xs:documentation>
    </xs:annotation>
    <xs:simpleType name="tConndir">
        <xs:restriction base="xs:normalizedString">
            <xs:enumeration value="horizontal"/>
            <xs:enumeration value="vertical"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:attribute name="x" type="xs:int"/>
    <xs:attribute name="y" type="xs:int"/>
    <xs:attribute name="dir" type="tConndir"/>
</xs:schema>
```

The following gives an SCL example using the coordinates. The transformer Baden220_132.T1 in this example will have the coordinates (1,10) relative to the substation. The bay D1Q1 of voltage level D1 will be located in the upper left corner of the substation layout.

Observe that this is a standardized extension, therefore the extension name (sxy) does not start with an e. For private extensions, it shall start with an e (see 8.3.5).

```
<?xml version="1.0"?>

<SCL xmlns="http://www.iec.ch/61850/2003/SCL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sxy="http://www.iec.ch/61850/2003/SCLcoordinates"

xsi:schemaLocation="http://www.iec.ch/61850/2003/SCL  SCL.xsd

                    http://www.iec.ch/61850/2003/SCLcoordinates SCL_Coordinates.xsd" version="2007" revision="A">

    <Header id="SCL Example T1-1" nameStructure="IEDName"/>

    <Substation name="Baden220_132" sxy:x="1" sxy:y="1" >

        <PowerTransformer name="T1" type="PTR" sxy:x="1" sxy:y="10" sxy:dir="horizontal">

            <TransformerWinding name="W1" type="PTW">

            </TransformerWinding>

            <TransformerWinding name="W2" type="PTW">

            </TransformerWinding>

        </PowerTransformer>

        <VoltageLevel name="D1" sxy:x="1" sxy:y="1">

            <Bay name="Q1" sxy:x="1" sxy:y="1" sxy:dir="horizontal"/>

        </VoltageLevel>

    </Substation>

</SCL>
```

Annex D
(informative)

Example

D.1 Example specification

D.1.1 General

An example based on the specification in I.1.3.2 of IEC 61850-5:2013 is given here. The naming of devices is, however, changed to conform to the IEC 81346 series. Although this example is not 100 % complete, it illustrates most of the SCL possibilities for system description, i.e. it is an SCD file.

D.1.2 Substation configuration

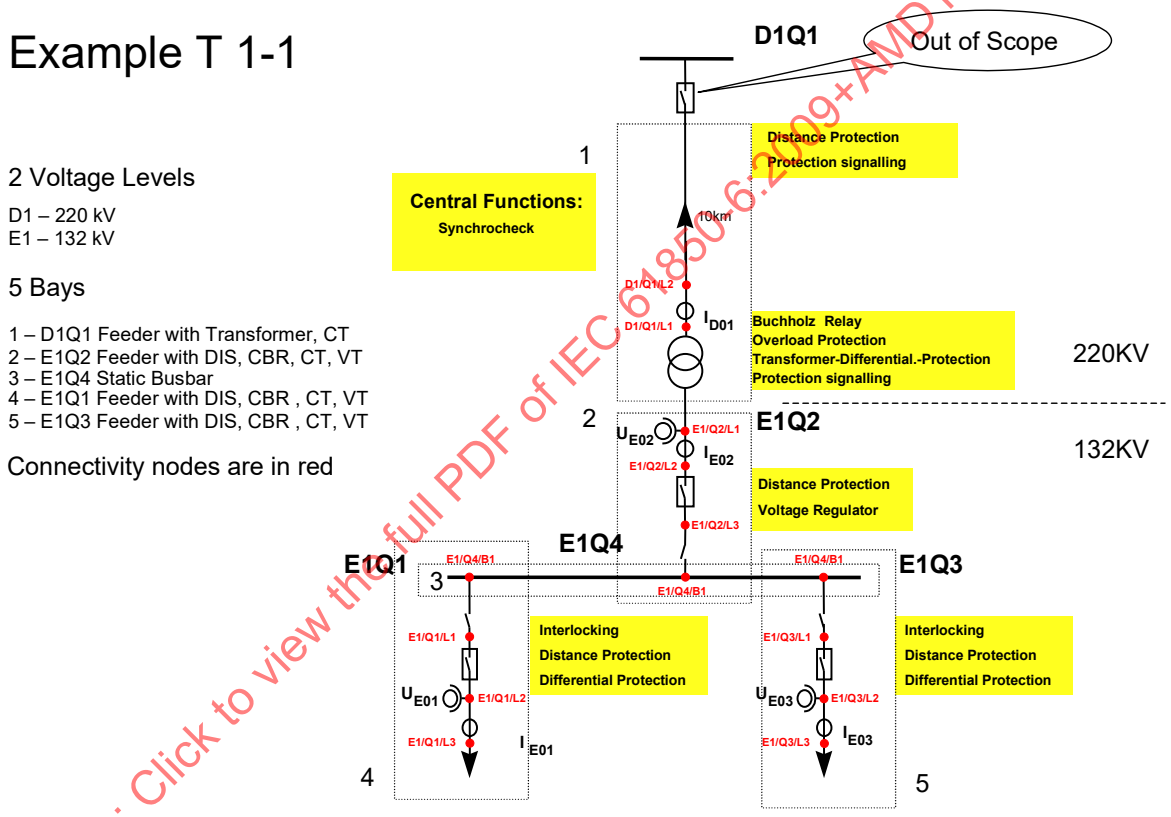


Figure D.1 – T1-1 Substation configuration

Figure D.1 shows the single line. The current infeed via D1Q1 to the transformer D1T1 is distributed at the lower voltage side to two lines E1Q1 and E1Q3. The circuit breaker in D1Q1 shall be out of the scope of the considered SA system.

D.1.3 Communication system configuration

Example T 1-1

Single communication bus

IEDs for:
Transformer.
Combined Bay Unit (Circuit Breaker, Disconnector, CT and VT).
Each Protection.
Central Functions.

No.	Name	ID
1	Dist	E1Q1BP3 (PDIS)
2	Difn	E1Q1BP2 (PDIF)
3	Dist	E1Q3BP3 (PDIS)
4	Difn	E1Q3BP2 (PDIF)
5	Dist	D1Q1BP3 (PDIS)
6	TDifn	D1Q1BP2 (PDIF)
7	Trafo	D1Q1SB1
8	LV Bay1	E1Q2SB1
9	LV Bay2	E1Q1SB1
10	LV Bay3	E1Q3SB1
11	Central	D1Q1SB4 (CILO, RSYN)

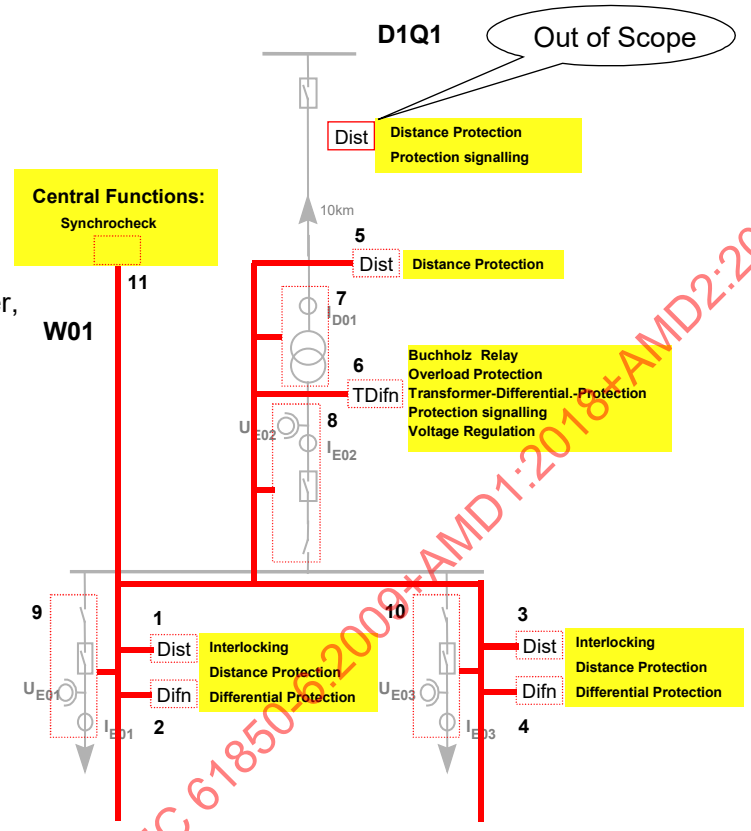


Figure D.2 – T1-1 Communication configuration

Figure D.2 shows the IEDs of the SA system, their allocation to the switch yard bays, their intended functionality and their communication connection by one single Subnetwork. What is not shown is the IED hosting the station level HMI, which might be a pure client.

D.1.4 Transformer IED

Figure D.3 illustrates the instantiated functionality for the transformer control IED as logical nodes.

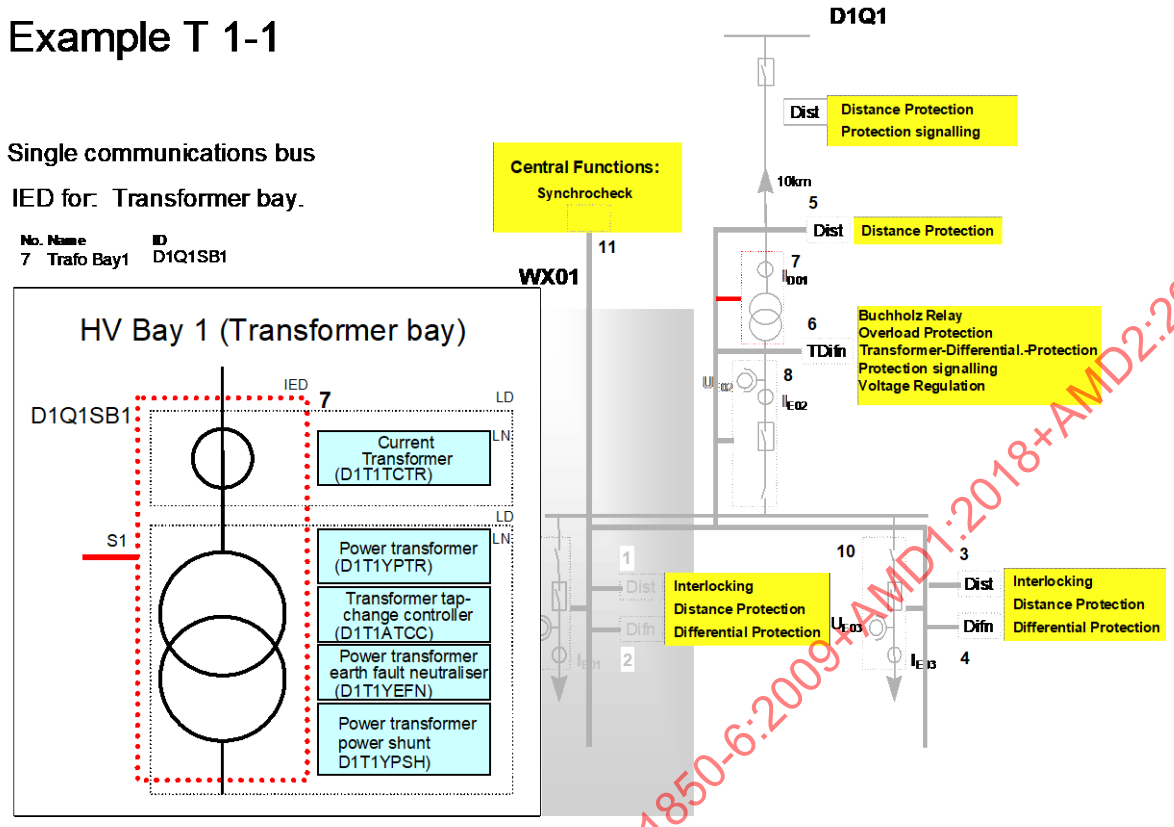


Figure D.3 – T1-1 Transformer bay

D.2 Example SCL file contents

In the SCL code component defined in 1.3, a syntactically correct example SCD file is provided as information, but not a fully completed SCD file for the example specification given above. For some IEDs, the server description is missing and naturally no data flow from or to these IEDs is specified. On the other hand, some logical nodes which should reside on these IEDs have been allocated to the substation section. Therefore, this file is not only incomplete but also invalid at application level. However, the two IEDs E1Q1SB1 and D1Q1SB4 and some data flow between them with GOOSE and SV is modelled, and the substation topology as such is complete with connection information. The Subnet definition is also complete, at least for the modelled data flow.

The example SCD is available as a code component at the TC 57 website with the schema code component. For more information about the SCL code component see Subclause 1.3.

Annex E
(informative)

SCL syntax: General XML schema definition

E.1 General

By using the mayIgnore rules any tool claiming conformance to this standard shall also accept valid SCL files according to the SCL version 2003 A and the problem (tissue) solutions before this version 2007 B. This annex E therefore contains a SCL schema which informally defines everything which shall be syntactically allowed to cover the language versions 2003 A and 2007 A and B. It has to be kept in mind, that this schema cannot be used as input check in general because it would surely fail for any follower SCL version, which shall be acceptable due to the mustUnderstand and mayIgnore rules.

The purpose of this annex is to simply give an idea of what a version 2007 tool has to accept as input, beneath the usage of mustUnderstand and mayIgnore rules.

E.2 Base types

SCL_BaseSimpleTypes.xsd

<CODE BEGINS>

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://www.iec.ch/61850/2003/SCL"
targetNamespace="http://www.iec.ch/61850/2003/SCL" elementFormDefault="qualified" attributeFormDefault="unqualified"
version="Mixed.2007B4">

        <xs:annotation>

                <xs:documentation xml:lang="en">

                        Informative general SCL schema for mixed systems version "2007" revision "B" release 4,  for
IEC 61850-6 Ed. 2.1.


                        COPYRIGHT (c) IEC, 2016. All rights reserved. Disclaimer: The IEC disclaims liability for any personal
injury, property or other damages of any nature whatsoever, whether special, indirect, consequential or compensatory, directly
or indirectly resulting from this software and the document upon which its methods are based, use of, or reliance upon.

                </xs:documentation>
        </xs:annotation>
        <xs:simpleType name="tConnectivityNodeReference">
                <xs:restriction base="xs:normalizedString">
                        <xs:pattern value=".+/.+(/.+)*"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tAnyName">
                <xs:restriction base="xs:normalizedString"/>
        </xs:simpleType>
```

```
<xs:simpleType name="tName">
        <xs:restriction base="tAnyName">
                <xs:minLength value="1"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tID">
        <xs:restriction base="xs:token">
                <xs:minLength value="1"/>
                <xs:maxLength value="255"/>
                <xs:pattern value="\S+"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tAcsiName">
        <xs:restriction base="xs:Name">
                <xs:pattern value="[A-Za-z][0-9A-Za-z_]*"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tRestrName1stU">
        <xs:restriction base="xs:Name">
                <xs:pattern value="[A-Z][0-9A-Za-z]*"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tRestrName1stL">
        <xs:restriction base="xs:Name">
                <xs:pattern value="[a-z][0-9A-Za-z]*"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tPAddr">
        <xs:restriction base="xs:normalizedString">
                <xs:minLength value="1"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tSclVersion">
        <xs:restriction base="tName">
                <xs:pattern value="2[0-2][0-9]{2}"/>
        </xs:restriction>
```

```
    </xs:simpleType>
    <xs:simpleType name="tSclRevision">
        <xs:restriction base="xs:Name">
            <xs:pattern value="[A-Z]"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tSclRelease">
        <xs:restriction base="xs:unsignedByte">
            <xs:minExclusive value="0"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tEmpty">
        <xs:restriction base="xs:normalizedString">
            <xs:maxLength value="0"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tIEDName">
        <xs:restriction base="tAcsiName">
            <xs:maxLength value="64"/>
            <xs:pattern value="[A-Za-z][0-9A-Za-z_]{0,2}"/>
            <xs:pattern value="[A-Za-z][0-9A-Za-z_]{4,63}"/>
            <xs:pattern value="[A-MO-Za-z][0-9A-Za-z_]{3}"/>
            <xs:pattern value="N[0-9A-Za-np-z_][0-9A-Za-z_]{2}"/>
            <xs:pattern value="No[0-9A-Za-mo-z_][0-9A-Za-z_]"/>
            <xs:pattern value="Non[0-9A-Za-df-z_]"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tIEDNameIsNone">
        <xs:restriction base="tAcsiName">
            <xs:pattern value="None"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tIEDNameOrNone">
        <xs:union memberTypes="tIEDName tIEDNameIsNone"/>
    </xs:simpleType>
    <xs:simpleType name="tOnlyRelativeIEDName">
```

```
        <xs:restriction base="xs:normalizedString">
            <xs:pattern value="&#x0040;"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tIEDNameOrRelative">
        <xs:union memberTypes="tIEDName tOnlyRelativeIEDName"/>
    </xs:simpleType>
    <xs:simpleType name="tLDName">
        <xs:restriction base="xs:normalizedString">
            <xs:maxLength value="64"/>
            <xs:pattern value="[A-Za-z][0-9A-Za-z_]*"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tLDInst">
        <xs:restriction base="xs:normalizedString">
            <xs:maxLength value="64"/>
            <xs:pattern value="[A-Za-z0-9][0-9A-Za-z_]*"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tLDInstOrEmpty">
        <xs:union memberTypes="tLDInst tEmpty"/>
    </xs:simpleType>
    <xs:simpleType name="tPrefix">
        <xs:restriction base="xs:normalizedString">
            <xs:maxLength value="11"/>
            <xs:pattern value="[A-Za-z][0-9A-Za-z_]*"/>
            <xs:pattern value=""/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tLNInst">
        <xs:restriction base="xs:normalizedString">
            <xs:pattern value="[0-9]{1,12}"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tLNInstOrEmpty">
        <xs:union memberTypes="tLNInst tEmpty"/>
```

```
        </xs:simpleType>
        <xs:simpleType name="tDataName">
            <xs:restriction base="tRestrName1stU">
                <xs:maxLength value="12"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tDataSetName">
            <xs:restriction base="tAcsiName">
                <xs:maxLength value="32"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tCBName">
            <xs:restriction base="tAcsiName">
                <xs:maxLength value="32"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tLogName">
            <xs:restriction base="tAcsiName">
                <xs:maxLength value="32"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tAccessPointName">
            <xs:restriction base="xs:normalizedString">
                <xs:pattern value="[A-Za-z0-9][0-9A-Za-z_]*"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tAssociationID">
            <xs:restriction base="xs:normalizedString">
                <xs:minLength value="1"/>
                <xs:pattern value="[0-9A-Za-z]+"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tVisibleBasicLatin">
            <xs:restriction base="xs:normalizedString">
                <xs:pattern value="[&#x0020;-&#x007E;]*"/>
            </xs:restriction>
```

```
</xs:simpleType>
<xs:simpleType name="tMessageID">
        <xs:restriction base="tVisibleBasicLatin">
                <xs:minLength value="1"/>
                <xs:maxLength value="129"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tFullAttributeName">
        <xs:restriction base="xs:normalizedString">
                <xs:pattern value="[a-zA-Z][a-zA-Z0-9]*(\([0-9]+\))?(\.[a-zA-Z][a-zA-Z0-9]*(\([0-9]+\))?)*"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tFullDOName">
        <xs:restriction base="xs:normalizedString">
                <xs:pattern value="[A-Z][0-9A-Za-z]{0,11}(\.[a-z][0-9A-Za-z]*(\([0-9]+\))?)?"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tSubDataName">
        <xs:restriction base="tRestrName1stL">
                <xs:minLength value="1"/>
                <xs:maxLength value="60"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tNamespaceName">
        <xs:restriction base="xs:normalizedString">
                <xs:pattern value="[&#x0020;-&#x007E;]+:20\d\d[A-Z]?"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tLineType">
        <xs:restriction base="xs:normalizedString">
                <xs:minLength value="1"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tProcessType">
        <xs:restriction base="xs:normalizedString">
                <xs:minLength value="1"/>
```

```
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tProcessName">
		<xs:restriction base="xs:normalizedString">
			<xs:pattern value=".+(/.+)*"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tEnumStringValue">
		<xs:restriction base="xs:normalizedString">
			<xs:maxLength value="127"/>
			<xs:pattern value="[\p{IsBasicLatin}\p{IsLatin-1Supplement}]*"/>
		</xs:restriction>
	</xs:simpleType>
</xs:schema>
```

<CODE ENDS>

SCL_Enums.xsd

<CODE BEGINS>

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:scl="http://www.iec.ch/61850/2003/SCL" xmlns="http://www.iec.ch/61850/2003/SCL"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.iec.ch/61850/2003/SCL"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="Mixed.2007B4">
	<xs:annotation>
		<xs:documentation xml:lang="en">
			Informative general SCL schema for mixed systems version "2007" revision "B" release 4,  for
IEC 61850-6 Ed. 2.1.

			COPYRIGHT (c) IEC, 2016. All rights reserved. Disclaimer: The IEC disclaims liability for any personal
injury, property or other damages of any nature whatsoever, whether special, indirect, consequential or compensatory, directly
or indirectly resulting from this software and the document upon which its methods are based, use of, or reliance upon.
		</xs:documentation>
	</xs:annotation>
	<xs:include schemaLocation="SCL_BaseSimpleTypes.xsd"/>
	<xs:simpleType name="tPredefinedPTypeEnum">
		<xs:restriction base="xs:Name">
			<xs:enumeration value="IP"/>
```

```
                <xs:enumeration value="IP-SUBNET"/>
                <xs:enumeration value="IP-GATEWAY"/>
                <xs:enumeration value="OSI-NSAP"/>
                <xs:enumeration value="OSI-TSEL"/>
                <xs:enumeration value="OSI-SSEL"/>
                <xs:enumeration value="OSI-PSEL"/>
                <xs:enumeration value="OSI-AP-Title"/>
                <xs:enumeration value="OSI-AP-Invoke"/>
                <xs:enumeration value="OSI-AE-Qualifier"/>
                <xs:enumeration value="OSI-AE-Invoke"/>
                <xs:enumeration value="MAC-Address"/>
                <xs:enumeration value="APPID"/>
                <xs:enumeration value="VLAN-PRIORITY"/>
                <xs:enumeration value="VLAN-ID"/>
                <xs:enumeration value="SNTP-Port"/>
                <xs:enumeration value="MMS-Port"/>
                <xs:enumeration value="DNSName"/>
                <xs:enumeration value="IPv6FlowLabel"/>
                <xs:enumeration value="IPv6ClassOfTraffic"/>
                <xs:enumeration value="C37-118-IP-Port"/>
                <xs:enumeration value="IP-UDP-PORT"/>
                <xs:enumeration value="IP-TCP-PORT"/>
                <xs:enumeration value="IPv6"/>
                <xs:enumeration value="IPv6-SUBNET"/>
                <xs:enumeration value="IPv6-GATEWAY"/>
                <xs:enumeration value="IPv6-IGMPv3Src"/>
                <xs:enumeration value="IP-IGMPv3Src"/>
                <xs:enumeration value="IP-ClassOfTraffic"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tExtensionPTypeEnum">
            <xs:restriction base="xs:normalizedString">
                <xs:pattern value="[A-Z][0-9A-Za-z\-]*"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tPTypeEnum">
```

```
            <xs:union memberTypes="tPredefinedPTypeEnum tExtensionPTypeEnum"/>
        </xs:simpleType>
        <xs:simpleType name="tPredefinedPTypePhysConnEnum">
            <xs:restriction base="xs:Name">
                <xs:enumeration value="Type"/>
                <xs:enumeration value="Plug"/>
                <xs:enumeration value="Cable"/>
                <xs:enumeration value="Port"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tPTypePhysConnEnum">
            <xs:union memberTypes="tPredefinedPTypePhysConnEnum tExtensionPTypeEnum"/>
        </xs:simpleType>
        <xs:simpleType name="tPredefinedAttributeNameEnum">
            <xs:restriction base="xs:Name">
                <xs:enumeration value="T"/>
                <xs:enumeration value="Test"/>
                <xs:enumeration value="Check"/>
                <xs:enumeration value="SIUnit"/>
                <xs:enumeration value="Oper"/>
                <xs:enumeration value="SBO"/>
                <xs:enumeration value="SBOw"/>
                <xs:enumeration value="Cancel"/>
                <xs:enumeration value="Addr"/>
                <xs:enumeration value="PRIORITY"/>
                <xs:enumeration value="VID"/>
                <xs:enumeration value="APPID"/>
                <xs:enumeration value="TransportInUse"/>
                <xs:enumeration value="IPClassOfTraffic"/>
                <xs:enumeration value="IPv6FlowLabel"/>
                <xs:enumeration value="IPAddressLength"/>
                <xs:enumeration value="IPAddress"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tExtensionAttributeNameEnum">
            <xs:restriction base="tRestrName1stL">
```

```
            <xs:maxLength value="60"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tAttributeNameEnum">
        <xs:union memberTypes="tPredefinedAttributeNameEnum tExtensionAttributeNameEnum"/>
    </xs:simpleType>
    <xs:simpleType name="tPredefinedCommonConductingEquipmentEnum">
        <xs:restriction base="xs:Name">
            <xs:enumeration value="CBR"/>
            <xs:enumeration value="DIS"/>
            <xs:enumeration value="VTR"/>
            <xs:enumeration value="CTR"/>
            <xs:enumeration value="GEN"/>
            <xs:enumeration value="CAP"/>
            <xs:enumeration value="REA"/>
            <xs:enumeration value="CON"/>
            <xs:enumeration value="MOT"/>
            <xs:enumeration value="EFN"/>
            <xs:enumeration value="PSH"/>
            <xs:enumeration value="BAT"/>
            <xs:enumeration value="BSH"/>
            <xs:enumeration value="CAB"/>
            <xs:enumeration value="GIL"/>
            <xs:enumeration value="LIN"/>
            <xs:enumeration value="RES"/>
            <xs:enumeration value="RRC"/>
            <xs:enumeration value="SAR"/>
            <xs:enumeration value="TCF"/>
            <xs:enumeration value="TCR"/>
            <xs:enumeration value="IFL"/>
            <xs:enumeration value="FAN"/>
            <xs:enumeration value="SCR"/>
            <xs:enumeration value="SMC"/>
            <xs:enumeration value="PMP"/>
        </xs:restriction>
    </xs:simpleType>
```

```
<xs:simpleType name="tExtensionEquipmentEnum">
<xs:restriction base="xs:Name">
<xs:minLength value="3"/>
<xs:pattern value="E[A-Z]*"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="tCommonConductingEquipmentEnum">
<xs:union memberTypes="tPredefinedCommonConductingEquipmentEnum tExtensionEquipmentEnum"/>
</xs:simpleType>
<xs:simpleType name="tPowerTransformerEnum">
<xs:restriction base="xs:Name">
<xs:enumeration value="PTR"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="tTransformerWindingEnum">
<xs:restriction base="xs:Name">
<xs:enumeration value="PTW"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="tPredefinedGeneralEquipmentEnum">
<xs:restriction base="xs:Name">
<xs:enumeration value="AXN"/>
<xs:enumeration value="BAT"/>
<xs:enumeration value="MOT"/>
<xs:enumeration value="FAN"/>
<xs:enumeration value="FIL"/>
<xs:enumeration value="PMP"/>
<xs:enumeration value="TNK"/>
<xs:enumeration value="VLV"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="tExtensionGeneralEquipmentEnum">
<xs:restriction base="xs:Name">
<xs:minLength value="3"/>
<xs:pattern value="E[A-Z]*"/>
</xs:restriction>
```

```
</xs:simpleType>
<xs:simpleType name="tGeneralEquipmentEnum">
        <xs:union memberTypes="tPredefinedGeneralEquipmentEnum tExtensionGeneralEquipmentEnum"/>
</xs:simpleType>
<xs:simpleType name="tServiceSettingsNoDynEnum">
        <xs:restriction base="xs:Name">
                <xs:enumeration value="Conf"/>
                <xs:enumeration value="Fix"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tServiceSettingsEnum">
        <xs:restriction base="xs:Name">
                <xs:enumeration value="Dyn"/>
                <xs:enumeration value="Conf"/>
                <xs:enumeration value="Fix"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tRedProtEnum">
        <xs:restriction base="xs:Name">
                <xs:enumeration value="none"/>
                <xs:enumeration value="hsr"/>
                <xs:enumeration value="prp"/>
                <xs:enumeration value="rstp"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tSMVDeliveryEnum">
        <xs:restriction base="xs:Name">
                <xs:enumeration value="unicast"/>
                <xs:enumeration value="multicast"/>
                <xs:enumeration value="both"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tPhaseEnum">
        <xs:restriction base="xs:Name">
                <xs:enumeration value="A"/>
                <xs:enumeration value="B"/>
```

```
                    <xs:enumeration value="C"/>
                    <xs:enumeration value="N"/>
                    <xs:enumeration value="all"/>
                    <xs:enumeration value="none"/>
                    <xs:enumeration value="AB"/>
                    <xs:enumeration value="BC"/>
                    <xs:enumeration value="CA"/>
                </xs:restriction>
            </xs:simpleType>
            <xs:simpleType name="tAuthenticationEnum">
                <xs:restriction base="xs:Name">
                    <xs:enumeration value="none"/>
                    <xs:enumeration value="password"/>
                    <xs:enumeration value="weak"/>
                    <xs:enumeration value="strong"/>
                    <xs:enumeration value="certificate"/>
                </xs:restriction>
            </xs:simpleType>
            <xs:simpleType name="tAssociationKindEnum">
                <xs:restriction base="xs:token">
                    <xs:enumeration value="pre-established"/>
                    <xs:enumeration value="predefined"/>
                </xs:restriction>
            </xs:simpleType>
            <xs:simpleType name="tLPHDEnum">
                <xs:restriction base="xs:Name">
                    <xs:enumeration value="LPHD"/>
                </xs:restriction>
            </xs:simpleType>
            <xs:simpleType name="tLLN0Enum">
                <xs:restriction base="xs:Name">
                    <xs:enumeration value="LLN0"/>
                </xs:restriction>
            </xs:simpleType>
            <xs:simpleType name="tSystemLNGroupEnum">
                <xs:restriction base="xs:Name">
```

```
            <xs:length value="4"/>
            <xs:pattern value="L[A-Z]*"/>
            <xs:pattern value="LLN0"/>
            <xs:enumeration value="LLN0"/>
            <xs:enumeration value="LPHD"/>
            <xs:enumeration value="LCCH"/>
            <xs:enumeration value="LGOS"/>
            <xs:enumeration value="LSVS"/>
            <xs:enumeration value="LTIM"/>
            <xs:enumeration value="LTMS"/>
            <xs:enumeration value="LTRK"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tDomainLNGroupAEnum">
        <xs:restriction base="xs:Name">
            <xs:length value="4"/>
            <xs:pattern value="A[A-Z]*"/>
            <xs:enumeration value="ANCR"/>
            <xs:enumeration value="ARCO"/>
            <xs:enumeration value="ARIS"/>
            <xs:enumeration value="ATCC"/>
            <xs:enumeration value="AVCO"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tDomainLNGroupCEnum">
        <xs:restriction base="xs:Name">
            <xs:length value="4"/>
            <xs:pattern value="C[A-Z]*"/>
            <xs:enumeration value="CALH"/>
            <xs:enumeration value="CCGR"/>
            <xs:enumeration value="CILO"/>
            <xs:enumeration value="CPOW"/>
            <xs:enumeration value="CSWI"/>
            <xs:enumeration value="CSYN"/>
        </xs:restriction>
    </xs:simpleType>
```

```
        <xs:simpleType name="tDomainLNGroupFEnum">
            <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="F[A-Z]*"/>
                <xs:enumeration value="FCNT"/>
                <xs:enumeration value="FCSD"/>
                <xs:enumeration value="FFIL"/>
                <xs:enumeration value="FLIM"/>
                <xs:enumeration value="FPID"/>
                <xs:enumeration value="FRMP"/>
                <xs:enumeration value="FSPT"/>
                <xs:enumeration value="FXOT"/>
                <xs:enumeration value="FXUT"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tDomainLNGroupGEnum">
            <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="G[A-Z]*"/>
                <xs:enumeration value="GAPC"/>
                <xs:enumeration value="GGIO"/>
                <xs:enumeration value="GLOG"/>
                <xs:enumeration value="GSAL"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tDomainLNGroupIEnum">
            <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="I[A-Z]*"/>
                <xs:enumeration value="IARC"/>
                <xs:enumeration value="IHMI"/>
                <xs:enumeration value="ISAF"/>
                <xs:enumeration value="ITCI"/>
                <xs:enumeration value="ITMI"/>
                <xs:enumeration value="ITPC"/>
            </xs:restriction>
```

```
</xs:simpleType>
<xs:simpleType name="tDomainLNGroupKEnum">
        <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="K[A-Z]*"/>
                <xs:enumeration value="KFAN"/>
                <xs:enumeration value="KFIL"/>
                <xs:enumeration value="KPMP"/>
                <xs:enumeration value="KTNK"/>
                <xs:enumeration value="KVLV"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tDomainLNGroupMEnum">
        <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="M[A-Z]*"/>
                <xs:enumeration value="MDIF"/>
                <xs:enumeration value="MENV"/>
                <xs:enumeration value="MFLK"/>
                <xs:enumeration value="MHAI"/>
                <xs:enumeration value="MHAN"/>
                <xs:enumeration value="MHYD"/>
                <xs:enumeration value="MMDC"/>
                <xs:enumeration value="MMET"/>
                <xs:enumeration value="MMTN"/>
                <xs:enumeration value="MMTR"/>
                <xs:enumeration value="MMXN"/>
                <xs:enumeration value="MMXU"/>
                <xs:enumeration value="MSQI"/>
                <xs:enumeration value="MSTA"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="tDomainLNGroupPEnum">
        <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="P[A-Z]*"/>
```

```
                <xs:enumeration value="PDIF"/>
                <xs:enumeration value="PDIR"/>
                <xs:enumeration value="PDIS"/>
                <xs:enumeration value="PDOP"/>
                <xs:enumeration value="PDUP"/>
                <xs:enumeration value="PFRC"/>
                <xs:enumeration value="PHAR"/>
                <xs:enumeration value="PHIZ"/>
                <xs:enumeration value="PIOC"/>
                <xs:enumeration value="PMRI"/>
                <xs:enumeration value="PMSS"/>
                <xs:enumeration value="POPF"/>
                <xs:enumeration value="PPAM"/>
                <xs:enumeration value="PRTR"/>
                <xs:enumeration value="PSCH"/>
                <xs:enumeration value="PSDE"/>
                <xs:enumeration value="PTEF"/>
                <xs:enumeration value="PTHF"/>
                <xs:enumeration value="PTOC"/>
                <xs:enumeration value="PTOF"/>
                <xs:enumeration value="PTOV"/>
                <xs:enumeration value="PTRC"/>
                <xs:enumeration value="PTTR"/>
                <xs:enumeration value="PTUC"/>
                <xs:enumeration value="PTUF"/>
                <xs:enumeration value="PTUV"/>
                <xs:enumeration value="PUPF"/>
                <xs:enumeration value="PVOC"/>
                <xs:enumeration value="PVPH"/>
                <xs:enumeration value="PZSU"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tDomainLNGroupQEnum">
            <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="Q[A-Z]*"/>
```

```
            <xs:enumeration value="QFVR"/>
            <xs:enumeration value="QITR"/>
            <xs:enumeration value="QIUB"/>
            <xs:enumeration value="QVTR"/>
            <xs:enumeration value="QVUB"/>
            <xs:enumeration value="QVVR"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tDomainLNGroupREnum">
        <xs:restriction base="xs:Name">
            <xs:length value="4"/>
            <xs:pattern value="R[A-Z]*"/>
            <xs:enumeration value="RADR"/>
            <xs:enumeration value="RBDR"/>
            <xs:enumeration value="RBRF"/>
            <xs:enumeration value="RDIR"/>
            <xs:enumeration value="RDRE"/>
            <xs:enumeration value="RDRS"/>
            <xs:enumeration value="RFLO"/>
            <xs:enumeration value="RMXU"/>
            <xs:enumeration value="RPSB"/>
            <xs:enumeration value="RREC"/>
            <xs:enumeration value="RSYN"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tDomainLNGroupSEnum">
        <xs:restriction base="xs:Name">
            <xs:length value="4"/>
            <xs:pattern value="S[A-Z]*"/>
            <xs:enumeration value="SARC"/>
            <xs:enumeration value="SCBR"/>
            <xs:enumeration value="SIMG"/>
            <xs:enumeration value="SIML"/>
            <xs:enumeration value="SLTC"/>
            <xs:enumeration value="SOPM"/>
            <xs:enumeration value="SPDC"/>
```

```
                <xs:enumeration value="SPTR"/>
                <xs:enumeration value="SSWI"/>
                <xs:enumeration value="STMP"/>
                <xs:enumeration value="SVBR"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tDomainLNGroupTEnum">
            <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="T[A-Z]*"/>
                <xs:enumeration value="TANG"/>
                <xs:enumeration value="TAXD"/>
                <xs:enumeration value="TCTR"/>
                <xs:enumeration value="TDST"/>
                <xs:enumeration value="TFLW"/>
                <xs:enumeration value="TFRQ"/>
                <xs:enumeration value="TGSN"/>
                <xs:enumeration value="THUM"/>
                <xs:enumeration value="TLVL"/>
                <xs:enumeration value="TMGF"/>
                <xs:enumeration value="TMVM"/>
                <xs:enumeration value="TPOS"/>
                <xs:enumeration value="TPRS"/>
                <xs:enumeration value="TRTN"/>
                <xs:enumeration value="TSND"/>
                <xs:enumeration value="TTMP"/>
                <xs:enumeration value="TTNS"/>
                <xs:enumeration value="TVBR"/>
                <xs:enumeration value="TVTR"/>
                <xs:enumeration value="TWPH"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tDomainLNGroupXEnum">
            <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="X[A-Z]*"/>
```

```
            <xs:enumeration value="XCBR"/>
            <xs:enumeration value="XSWI"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tDomainLNGroupYEnum">
        <xs:restriction base="xs:Name">
            <xs:length value="4"/>
            <xs:pattern value="Y[A-Z]*"/>
            <xs:enumeration value="YEFN"/>
            <xs:enumeration value="YLTC"/>
            <xs:enumeration value="YPSH"/>
            <xs:enumeration value="YPTR"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tDomainLNGroupZEnum">
        <xs:restriction base="xs:Name">
            <xs:length value="4"/>
            <xs:pattern value="Z[A-Z]*"/>
            <xs:enumeration value="ZAXN"/>
            <xs:enumeration value="ZBAT"/>
            <xs:enumeration value="ZBSH"/>
            <xs:enumeration value="ZCAB"/>
            <xs:enumeration value="ZCAP"/>
            <xs:enumeration value="ZCON"/>
            <xs:enumeration value="ZGEN"/>
            <xs:enumeration value="ZGIL"/>
            <xs:enumeration value="ZLIN"/>
            <xs:enumeration value="ZMOT"/>
            <xs:enumeration value="ZREA"/>
            <xs:enumeration value="ZRES"/>
            <xs:enumeration value="ZRRC"/>
            <xs:enumeration value="ZSAR"/>
            <xs:enumeration value="ZSCR"/>
            <xs:enumeration value="ZSMC"/>
            <xs:enumeration value="ZTCF"/>
            <xs:enumeration value="ZTCR"/>
```

```
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tDomainLNEnum">
            <xs:union memberTypes="tDomainLNGroupAEnum tDomainLNGroupCEnum tDomainLNGroupFEnum
tDomainLNGroupGEnum tDomainLNGroupIEnum tDomainLNGroupKEnum tDomainLNGroupMEnum tDomainLNGroupPEnum
tDomainLNGroupQEnum tDomainLNGroupREnum tDomainLNGroupSEnum tDomainLNGroupTEnum tDomainLNGroupXEnum
tDomainLNGroupYEnum tDomainLNGroupZEnum"/>
        </xs:simpleType>
        <xs:simpleType name="tPredefinedLNClassEnum">
            <xs:union memberTypes="tSystemLNGroupEnum tDomainLNEnum"/>
        </xs:simpleType>
        <xs:simpleType name="tExtensionLNClassEnum">
            <xs:restriction base="xs:Name">
                <xs:length value="4"/>
                <xs:pattern value="[A-Z]+"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tLNClassEnum">
            <xs:union memberTypes="tPredefinedLNClassEnum tExtensionLNClassEnum"/>
        </xs:simpleType>
        <xs:simpleType name="tPredefinedCDCEnum">
            <xs:restriction base="xs:Name">
                <xs:enumeration value="SPS"/>
                <xs:enumeration value="DPS"/>
                <xs:enumeration value="INS"/>
                <xs:enumeration value="ENS"/>
                <xs:enumeration value="ACT"/>
                <xs:enumeration value="ACD"/>
                <xs:enumeration value="SEC"/>
                <xs:enumeration value="BCR"/>
                <xs:enumeration value="HST"/>
                <xs:enumeration value="VSS"/>
                <xs:enumeration value="MV"/>
                <xs:enumeration value="CMV"/>
                <xs:enumeration value="SAV"/>
                <xs:enumeration value="WYE"/>
                <xs:enumeration value="DEL"/>
                <xs:enumeration value="SEQ"/>
```

```
<xs:enumeration value="HMV"/>
<xs:enumeration value="HWYE"/>
<xs:enumeration value="HDEL"/>
<xs:enumeration value="SPC"/>
<xs:enumeration value="DPC"/>
<xs:enumeration value="INC"/>
<xs:enumeration value="ENC"/>
<xs:enumeration value="BSC"/>
<xs:enumeration value="ISC"/>
<xs:enumeration value="APC"/>
<xs:enumeration value="BAC"/>
<xs:enumeration value="SPG"/>
<xs:enumeration value="ING"/>
<xs:enumeration value="ENG"/>
<xs:enumeration value="ORG"/>
<xs:enumeration value="TSG"/>
<xs:enumeration value="CUG"/>
<xs:enumeration value="VSG"/>
<xs:enumeration value="ASG"/>
<xs:enumeration value="CURVE"/>
<xs:enumeration value="CSG"/>
<xs:enumeration value="DPL"/>
<xs:enumeration value="LPL"/>
<xs:enumeration value="CSD"/>
<xs:enumeration value="CST"/>
<xs:enumeration value="BTS"/>
<xs:enumeration value="UTS"/>
<xs:enumeration value="LTS"/>
<xs:enumeration value="GTS"/>
<xs:enumeration value="MTS"/>
<xs:enumeration value="NTS"/>
<xs:enumeration value="STS"/>
<xs:enumeration value="CTS"/>
<xs:enumeration value="OTS"/>
<xs:enumeration value="VSD"/>
<xs:enumeration value="ORS"/>
```

```
                <xs:enumeration value="TCS"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tExtensionCDCEnum">
            <xs:restriction base="xs:Name">
                <xs:minLength value="1"/>
                <xs:maxLength value="5"/>
                <xs:pattern value="[A-Za-z]+"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tCDCEnum">
            <xs:union memberTypes="tPredefinedCDCEnum tExtensionCDCEnum"/>
        </xs:simpleType>
        <xs:simpleType name="tFCEnum">
            <xs:restriction base="xs:Name">
                <xs:enumeration value="ST"/>
                <xs:enumeration value="MX"/>
                <xs:enumeration value="CO"/>
                <xs:enumeration value="SP"/>
                <xs:enumeration value="SG"/>
                <xs:enumeration value="SE"/>
                <xs:enumeration value="SV"/>
                <xs:enumeration value="CF"/>
                <xs:enumeration value="DC"/>
                <xs:enumeration value="EX"/>
                <xs:enumeration value="SR"/>
                <xs:enumeration value="BL"/>
                <xs:enumeration value="OR"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tPredefinedBasicTypeEnum">
            <xs:restriction base="xs:Name">
                <xs:enumeration value="BOOLEAN"/>
                <xs:enumeration value="INT8"/>
                <xs:enumeration value="INT16"/>
                <xs:enumeration value="INT24"/>
```

```
            <xs:enumeration value="INT32"/>
            <xs:enumeration value="INT64"/>
            <xs:enumeration value="INT128"/>
            <xs:enumeration value="INT8U"/>
            <xs:enumeration value="INT16U"/>
            <xs:enumeration value="INT24U"/>
            <xs:enumeration value="INT32U"/>
            <xs:enumeration value="FLOAT32"/>
            <xs:enumeration value="FLOAT64"/>
            <xs:enumeration value="Enum"/>
            <xs:enumeration value="Dbpos"/>
            <xs:enumeration value="Tcmd"/>
            <xs:enumeration value="Quality"/>
            <xs:enumeration value="Timestamp"/>
            <xs:enumeration value="VisString32"/>
            <xs:enumeration value="VisString64"/>
            <xs:enumeration value="VisString65"/>
            <xs:enumeration value="VisString129"/>
            <xs:enumeration value="VisString255"/>
            <xs:enumeration value="Octet64"/>
            <xs:enumeration value="Unicode255"/>
            <xs:enumeration value="Struct"/>
            <xs:enumeration value="EntryTime"/>
            <xs:enumeration value="Check"/>
            <xs:enumeration value="ObjRef"/>
            <xs:enumeration value="Currency"/>
            <xs:enumeration value="PhyComAddr"/>
            <xs:enumeration value="TrgOps"/>
            <xs:enumeration value="OptFlds"/>
            <xs:enumeration value="SvOptFlds"/>
            <xs:enumeration value="LogOptFlds"/>
            <xs:enumeration value="EntryID"/>
            <xs:enumeration value="Octet6"/>
            <xs:enumeration value="Octet16"/>
            <!-- for 61850-8-1 Edition 2.1 -->
        </xs:restriction>
```

```
    </xs:simpleType>
    <xs:simpleType name="tExtensionBasicTypeEnum">
        <xs:restriction base="xs:Name">
            <xs:pattern value="[A-Z][0-9A-Za-z]*"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tBasicTypeEnum">
        <xs:union memberTypes="tPredefinedBasicTypeEnum tExtensionBasicTypeEnum"/>
    </xs:simpleType>
    <xs:simpleType name="tValKindEnum">
        <xs:restriction base="xs:Name">
            <xs:enumeration value="Spec"/>
            <xs:enumeration value="Conf"/>
            <xs:enumeration value="RO"/>
            <xs:enumeration value="Set"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tGSEControlTypeEnum">
        <xs:restriction base="xs:Name">
            <xs:enumeration value="GSSE"/>
            <xs:enumeration value="GOOSE"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tUnitMultiplierEnum">
        <xs:restriction base="xs:normalizedString">
            <xs:enumeration value=""/>
            <xs:enumeration value="m"/>
            <xs:enumeration value="k"/>
            <xs:enumeration value="M"/>
            <xs:enumeration value="mu"/>
            <xs:enumeration value="y"/>
            <xs:enumeration value="z"/>
            <xs:enumeration value="a"/>
            <xs:enumeration value="f"/>
            <xs:enumeration value="p"/>
            <xs:enumeration value="n"/>
```

```
            <xs:enumeration value="c"/>
            <xs:enumeration value="d"/>
            <xs:enumeration value="da"/>
            <xs:enumeration value="h"/>
            <xs:enumeration value="G"/>
            <xs:enumeration value="T"/>
            <xs:enumeration value="P"/>
            <xs:enumeration value="E"/>
            <xs:enumeration value="Z"/>
            <xs:enumeration value="Y"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tRightEnum">
        <xs:restriction base="xs:normalizedString">
            <xs:enumeration value="full"/>
            <xs:enumeration value="fix"/>
            <xs:enumeration value="dataflow"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tSDOCount">
        <xs:union memberTypes="xs:unsignedInt tRestrName1stL"/>
    </xs:simpleType>
    <xs:simpleType name="tDACount">
        <xs:union memberTypes="xs:unsignedInt tAttributeNameEnum"/>
    </xs:simpleType>
    <xs:simpleType name="tSmpMod">
        <xs:restriction base="xs:normalizedString">
            <xs:enumeration value="SmpPerPeriod"/>
            <xs:enumeration value="SmpPerSec"/>
            <xs:enumeration value="SecPerSmp"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="tPredefinedPhysConnTypeEnum">
        <xs:restriction base="xs:normalizedString">
            <xs:enumeration value="Connection"/>
            <xs:enumeration value="RedConn"/>
```

```
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tExtensionPhysConnTypeEnum">
            <xs:restriction base="xs:normalizedString">
                <xs:pattern value="[A-Z][0-9A-Za-z\-]*"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tPhysConnTypeEnum">
            <xs:union memberTypes="tPredefinedPhysConnTypeEnum tExtensionPhysConnTypeEnum"/>
        </xs:simpleType>
        <xs:simpleType name="tServiceType">
            <xs:restriction base="xs:Name">
                <xs:enumeration value="Poll"/>
                <xs:enumeration value="Report"/>
                <xs:enumeration value="GOOSE"/>
                <xs:enumeration value="SMV"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="tPredefinedTypeOfSecurityEnum">
            <xs:restriction base="xs:normalizedString">
                <xs:enumeration value="None"/>
                <xs:enumeration value="Signature"/>
                <xs:enumeration value="SignatureAndEncryption"/>
            </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

<CODE ENDS>

SCL_BaseTypes.xsd

<CODE BEGINS>

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:scl="http://www.iec.ch/61850/2003/SCL" xmlns="http://www.iec.ch/61850/2003/SCL"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.iec.ch/61850/2003/SCL"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="Mixed.2007B4">

    <xs:annotation>
```

```
        <xs:documentation xml:lang="en">
            Informative general SCL schema for mixed systems version "2007" revision "B" release 4,  for IEC 61850-6 Ed. 2.1.
```



```
        </xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="SCL_Enums.xsd"/>
    <xs:attributeGroup name="agDesc">
        <xs:attribute name="desc" type="xs:normalizedString" use="optional" default=""/>
    </xs:attributeGroup>
    <xs:complexType name="tBaseElement" abstract="true">
        <xs:sequence>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Text" type="tText" minOccurs="0"/>
            <xs:element name="Private" type="tPrivate" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="tUnNaming" abstract="true">
        <xs:complexContent>
            <xs:extension base="tBaseElement">
                <xs:attributeGroup ref="agDesc"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tNaming" abstract="true">
        <xs:complexContent>
            <xs:extension base="tBaseElement">
                <xs:attribute name="name" type="tName" use="required"/>
                <xs:attributeGroup ref="agDesc"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tIDNaming" abstract="true">
        <xs:complexContent>
```

```
            <xs:extension base="tBaseElement">
                <xs:attribute name="id" type="tID" use="required"/>
                <xs:attributeGroup ref="agDesc"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tAnyContentFromOtherNamespace" abstract="true" mixed="true">
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
            <xs:any namespace="##other" processContents="lax"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="tText" mixed="true">
        <xs:complexContent>
            <xs:extension base="tAnyContentFromOtherNamespace">
                <xs:attribute name="source" type="xs:anyURI" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tPrivate" mixed="true">
        <xs:complexContent>
            <xs:extension base="tAnyContentFromOtherNamespace">
                <xs:attribute name="type" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:normalizedString">
                            <xs:minLength value="1"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="source" type="xs:anyURI" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tHeader">
        <xs:sequence>
            <xs:element name="Text" type="tText" minOccurs="0"/>
```

```
        <xs:element name="History" minOccurs="0">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="Hitem" type="tHitem" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="id" type="xs:normalizedString" use="required"/>
    <xs:attribute name="version" type="xs:normalizedString" use="optional"/>
    <xs:attribute name="revision" type="xs:normalizedString" use="optional" default=""/>
    <xs:attribute name="toolID" type="xs:normalizedString" use="optional"/>
    <xs:attribute name="nameStructure" use="optional" default="IEDName">
        <xs:simpleType>
            <xs:restriction base="xs:Name">
                <xs:enumeration value="IEDName"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
<xs:complexType name="tHitem" mixed="true">
    <xs:complexContent>
        <xs:extension base="tAhyContentFromOtherNamespace">
            <xs:attribute name="version" type="xs:normalizedString" use="required"/>
            <xs:attribute name="revision" type="xs:normalizedString" use="required"/>
            <xs:attribute name="when" type="xs:normalizedString" use="required"/>
            <xs:attribute name="who" type="xs:normalizedString"/>
            <xs:attribute name="what" type="xs:normalizedString"/>
            <xs:attribute name="why" type="xs:normalizedString"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tVal">
    <xs:simpleContent>
        <xs:extension base="xs:normalizedString">
            <xs:attribute name="sGroup" type="xs:unsignedInt" use="optional"/>
```

```
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="tValueWithUnit">
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attribute name="unit" type="xs:token" use="required"/>
                <xs:attribute name="multiplier" type="tUnitMultiplierEnum" use="optional" default=""/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="tVoltage">
        <xs:simpleContent>
            <xs:restriction base="tValueWithUnit">
                <xs:attribute name="unit" type="xs:token" use="required" fixed="V"/>
                <xs:attribute name="multiplier" type="tUnitMultiplierEnum" use="optional" default=""/>
            </xs:restriction>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="tDurationInSec">
        <xs:simpleContent>
            <xs:restriction base="tValueWithUnit">
                <xs:attribute name="unit" type="xs:token" use="required" fixed="s"/>
                <xs:attribute name="multiplier" type="tUnitMultiplierEnum" use="optional" default=""/>
            </xs:restriction>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="tDurationInMilliSec">
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attribute name="unit" type="xs:token" use="optional" fixed="s"/>
                <xs:attribute name="multiplier" type="tUnitMultiplierEnum" use="optional" fixed="m"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="tBitRateInMbPerSec">
```

```
        <xs:simpleContent>
            <xs:extension base="xs:decimal">
                <xs:attribute name="unit" type="xs:normalizedString" use="optional" fixed="b/s"/>
                <xs:attribute name="multiplier" type="tUnitMultiplierEnum" use="optional" fixed="M"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

<CODE ENDS>

E.3 Substation syntax

Identical to Clause A.2 (except version identification: schema version Mixed.2007B4).

E.4 Data type templates

Identical to Clause A.3 (except version identification: schema version Mixed.2007B4).

E.5 IED capabilities and structure

SCL_IED.xsd

This file contains most of the differences compared to the 2007B4 version. For backward compatibility it allows deprecated options, and sets defaults to instances from the 2003 A version.

<CODE BEGINS>

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:scl="http://www.iec.ch/61850/2003/SCL" xmlns="http://www.iec.ch/61850/2003/SCL"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.iec.ch/61850/2003/SCL"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="Mixed.2007B4">
    <xs:annotation>
        <xs:documentation xml:lang="en">
            Informative general SCL schema for mixed systems version "2007" revision "B" release 4,  for IEC 61850-6 Ed. 2.1.

            COPYRIGHT (c) IEC, 2016. All rights reserved. Disclaimer: The IEC disclaims liability for any personal injury,
property or other damages of any nature whatsoever, whether special, indirect, consequential or compensatory, directly or
indirectly resulting from this software and the document upon which its methods are based, use of, or reliance upon.
        </xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="SCL_BaseTypes.xsd"/>
    <xs:attributeGroup name="agAuthentication">
```

```
        <xs:attribute name="none" type="xs:boolean" use="optional" default="true"/>
        <xs:attribute name="password" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="weak" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="strong" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="certificate" type="xs:boolean" use="optional" default="false"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="agSmvOpts">
        <xs:attribute name="refreshTime" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="sampleSynchronized" type="xs:boolean" use="optional" fixed="true"/>
        <xs:attribute name="sampleRate" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="dataSet" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="security" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="dataRef" type="xs:boolean" use="optional"/>
        <xs:attribute name="timestamp" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="synchSourceId" type="xs:boolean" use="optional" default="false"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="agOptFields">
        <xs:attribute name="seqNum" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="timeStamp" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="dataSet" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="reasonCode" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="dataRef" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="entryID" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="configRef" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="bufOvfl" type="xs:boolean" use="optional" default="true"/>
        <xs:attribute name="segmentation" type="xs:boolean" use="optional" default="false"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="agLDRef">
        <xs:attributeGroup ref="scl:agDesc"/>
        <xs:attribute name="iedName" type="tIEDName" use="required"/>
        <xs:attribute name="ldInst" type="tLDInst" use="required"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="agLNRef">
        <xs:attributeGroup ref="agLDRef"/>
        <xs:attribute name="prefix" type="tPrefix" use="optional" default=""/>
        <xs:attribute name="lnClass" type="tLNClassEnum" use="required"/>
```

```
    <xs:attribute name="lnInst" type="tLNInstOrEmpty" use="required"/>
</xs:attributeGroup>
<xs:complexType name="tIED">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Services" type="tServices" minOccurs="0"/>
                <xs:element name="AccessPoint" type="tAccessPoint" maxOccurs="unbounded">
                    <xs:unique name="uniqueLNInAccessPoint">
                        <xs:selector xpath="./scl:LN"/>
                        <xs:field xpath="@inst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="KDC" type="tKDC" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="name" type="tIEDName" use="required"/>
            <xs:attribute name="type" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="manufacturer" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="configVersion" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="originalSclVersion" type="tSclVersion" use="optional" default="2003"/>
            <xs:attribute name="originalSclRevision" type="tSclRevision" use="optional" default="A"/>
            <xs:attribute name="originalSclRelease" type="tSclRelease" use="optional" default="1"/>
            <xs:attribute name="engRight" type="tRightEnum" use="optional" default="full"/>
            <xs:attribute name="owner" type="xs:normalizedString" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tServices">
    <xs:all>
        <xs:element name="DynAssociation" type="scl:tServiceWithOptionalMax" minOccurs="0"/>
        <xs:element name="SettingGroups" type="scl:tSettingGroups" minOccurs="0"/>
        <xs:element name="GetDirectory" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="GetDataObjectDefinition" type="scl:tServiceYesNo" minOccurs="0"/>
        <xs:element name="DataObjectDirectory" type="scl:tServiceYesNo" minOccurs="0"/>
```

```
            <xs:element name="GetDataSetValue" type="scl:tServiceYesNo" minOccurs="0"/>
            <xs:element name="SetDataSetValue" type="scl:tServiceYesNo" minOccurs="0"/>
            <xs:element name="DataSetDirectory" type="scl:tServiceYesNo" minOccurs="0"/>
            <xs:element name="ConfDataSet" type="scl:tServiceForConfDataSet" minOccurs="0"/>
            <xs:element name="DynDataSet" type="scl:tServiceWithMaxAndMaxAttributes" minOccurs="0"/>
            <xs:element name="ReadWrite" type="scl:tServiceYesNo" minOccurs="0"/>
            <xs:element name="TimerActivatedControl" type="scl:tServiceYesNo" minOccurs="0"/>
            <xs:element name="ConfReportControl" type="scl:tServiceConfReportControl" minOccurs="0"/>
            <xs:element name="GetCBValues" type="scl:tServiceYesNo" minOccurs="0"/>
            <xs:element name="ConfLogControl" type="scl:tServiceWithMaxNonZero" minOccurs="0"/>
            <xs:element name="ReportSettings" type="scl:tReportSettings" minOccurs="0"/>
            <xs:element name="LogSettings" type="scl:tLogSettings" minOccurs="0"/>
            <xs:element name="GSESettings" type="scl:tGSESettings" minOccurs="0"/>
            <xs:element name="SMVSettings" type="scl:tSMVSettings" minOccurs="0"/>
            <xs:element name="GSEDir" type="scl:tServiceYesNo" minOccurs="0"/>
            <xs:element name="GOOSE" type="scl:tGOOSEcapabilities" minOccurs="0"/>
            <xs:element name="GSSE" type="scl:tServiceWithMax" minOccurs="0"/>
            <xs:element name="SMVsc" type="scl:tSMVsc" minOccurs="0"/>
            <xs:element name="FileHandling" type="scl:tFileHandling" minOccurs="0"/>
            <xs:element name="ConfLNs" type="scl:tConfLNs" minOccurs="0"/>
            <xs:element name="ClientServices" type="scl:tClientServices" minOccurs="0"/>
            <xs:element name="ConfLdName" type="scl:tServiceYesNo" minOccurs="0"/>
            <xs:element name="SupSubscription" type="scl:tSupSubscription" minOccurs="0"/>
            <xs:element name="ConfSigRef" type="scl:tServiceWithMaxNonZero" minOccurs="0"/>
            <xs:element name="ValueHandling" type="scl:tValueHandling" minOccurs="0"/>
            <xs:element name="RedProt" type="scl:tRedProt" minOccurs="0"/>
            <xs:element name="TimeSyncProt" type="scl:tTimeSyncProt" minOccurs="0"/>
            <xs:element name="CommProt" type="scl:tCommProt" minOccurs="0"/>
        </xs:all>
        <xs:attribute name="nameLength" use="optional" default="32">
            <xs:simpleType>
                <xs:restriction base="xs:token">
                    <xs:pattern value="32"/>
                    <xs:pattern value="64"/>
                    <xs:pattern value="6[5-9]"/>
                    <xs:pattern value="[7-9]\d"/>
```

```
            <xs:pattern value="[1-9]\d\d+"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
    <xs:complexType name="tAccessPoint">
      <xs:complexContent>
        <xs:extension base="tUnNaming">
          <xs:sequence>
            <xs:choice minOccurs="0">
              <xs:element name="Server" type="scl:tServer">
                <xs:unique name="uniqueAssociationInServer">
                  <xs:selector xpath="./scl:Association"/>
                  <xs:field xpath="@associationID"/>
                </xs:unique>
              </xs:element>
              <xs:element ref="scl:LN" maxOccurs="unbounded"/>
              <xs:element name="ServerAt" type="tServerAt"/>
            </xs:choice>
            <xs:element name="Services" type="scl:tServices" minOccurs="0"/>
            <xs:element name="GOOSESecurity" type="tCertificate" minOccurs="0" maxOccurs="7"/>
            <xs:element name="SMVSecurity" type="tCertificate" minOccurs="0" maxOccurs="7"/>
          </xs:sequence>
          <xs:attribute name="name" type="tAccessPointName" use="required"/>
          <xs:attribute name="router" type="xs:boolean" use="optional" default="false"/>
          <xs:attribute name="clock" type="xs:boolean" use="optional" default="false"/>
          <xs:attribute name="kdc" type="xs:boolean" use="optional" default="false"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tCertificate">
      <xs:complexContent>
        <xs:extension base="tNaming">
          <xs:sequence>
            <xs:element name="Subject" type="tCert"/>
            <xs:element name="IssuerName" type="tCert"/>
```

```
            </xs:sequence>
            <xs:attribute name="xferNumber" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="serialNumber" use="required">
              <xs:simpleType>
                <xs:restriction base="xs:normalizedString">
                  <xs:minLength value="1"/>
                  <xs:pattern value="[0-9]+"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <xs:complexType name="tCert">
        <xs:attribute name="commonName" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:normalizedString">
              <xs:minLength value="4"/>
              <xs:pattern value="none"/>
              <xs:pattern value="CN=.+"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="idHierarchy" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:normalizedString">
              <xs:minLength value="1"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
      </xs:complexType>
      <xs:complexType name="tServerAt">
        <xs:complexContent>
          <xs:extension base="tUnNaming">
            <xs:attribute name="apName" type="tAccessPointName" use="required"/>
            <xs:attribute name="apUuid" type="tUUIDAttribute" use="optional"/>
```

```
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tServer">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Authentication">
                    <xs:complexType>
                        <xs:attributeGroup ref="agAuthentication"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="LDevice" type="tLDevice" maxOccurs="unbounded">
                    <xs:unique name="uniqueLNInLDevice">
                        <xs:selector xpath="./scl:LN"/>
                        <xs:field xpath="@inst"/>
                        <xs:field xpath="@lnClass"/>
                        <xs:field xpath="@prefix"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="Association" type="tAssociation" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="timeout" type="xs:unsignedInt" use="optional" default="30"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tLDevice">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element ref="LN0"/>
                <xs:element ref="LN" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="AccessControl" type="tAccessControl" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="inst" type="tLDInst" use="required"/>
            <xs:attribute name="ldName" type="tLDName" use="optional"/>
```

```
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tAccessControl" mixed="true">
        <xs:complexContent>
            <xs:extension base="tAnyContentFromOtherNamespace"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tAssociation">
        <xs:attributeGroup ref="agLNRef"/>
        <xs:attribute name="kind" type="tAssociationKindEnum" use="required"/>
        <xs:attribute name="associationID" type="tAssociationID" use="optional"/>
    </xs:complexType>
    <xs:element name="LN0">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="tLN0"/>
            </xs:complexContent>
        </xs:complexType>
        <xs:unique name="uniqueReportControlInLN0">
            <xs:selector xpath="./scl:ReportControl"/>
            <xs:field xpath="@name"/>
        </xs:unique>
        <xs:unique name="uniqueLogControlInLN0">
            <xs:selector xpath="./scl:LogControl"/>
            <xs:field xpath="@name"/>
        </xs:unique>
        <xs:unique name="uniqueGSEControlInLN0">
            <xs:selector xpath="./scl:GSEControl"/>
            <xs:field xpath="@name"/>
        </xs:unique>
        <xs:unique name="uniqueSampledValueControlInLN0">
            <xs:selector xpath="./scl:SampledValueControl"/>
            <xs:field xpath="@name"/>
        </xs:unique>
        <xs:key name="DataSetKeyLN0">
```

```
        <xs:selector xpath="./scl:DataSet"/>
        <xs:field xpath="@name"/>
      </xs:key>
      <xs:keyref name="ref2DataSetReportLN0" refer="DataSetKeyLN0">
        <xs:selector xpath="./scl:ReportControl"/>
        <xs:field xpath="@datSet"/>
      </xs:keyref>
      <xs:keyref name="ref2DataSetLogLN0" refer="DataSetKeyLN0">
        <xs:selector xpath="./scl:LogControl"/>
        <xs:field xpath="@datSet"/>
      </xs:keyref>
      <xs:keyref name="ref2DataSetGSELN0" refer="DataSetKeyLN0">
        <xs:selector xpath="./scl:GSEControl"/>
        <xs:field xpath="@datSet"/>
      </xs:keyref>
      <xs:keyref name="ref2DataSetSVLN0" refer="DataSetKeyLN0">
        <xs:selector xpath="./scl:SampledValueControl"/>
        <xs:field xpath="@datSet"/>
      </xs:keyref>
      <xs:unique name="uniqueDOIinLN0">
        <xs:selector xpath="./scl:DOI"/>
        <xs:field xpath="@name"/>
      </xs:unique>
      <xs:unique name="uniqueLogInLN0">
        <xs:selector xpath="./scl:Log"/>
        <xs:field xpath="@name"/>
      </xs:unique>
    </xs:element>
    <xs:element name="LN" type="tLN">
      <xs:unique name="uniqueReportControlInLN">
        <xs:selector xpath="./scl:ReportControl"/>
        <xs:field xpath="@name"/>
      </xs:unique>
      <xs:unique name="uniqueLogControlInLN">
        <xs:selector xpath="./scl:LogControl"/>
        <xs:field xpath="@name"/>
```

```
        </xs:unique>
        <xs:key name="DataSetKeyInLN">
            <xs:selector xpath="./scl:DataSet"/>
            <xs:field xpath="@name"/>
        </xs:key>
        <xs:keyref name="ref2DataSetReport" refer="DataSetKeyInLN">
            <xs:selector xpath="./scl:ReportControl"/>
            <xs:field xpath="@datSet"/>
        </xs:keyref>
        <xs:keyref name="ref2DataSetLog" refer="DataSetKeyInLN">
            <xs:selector xpath="./scl:LogControl"/>
            <xs:field xpath="@datSet"/>
        </xs:keyref>
        <xs:unique name="uniqueDOIinLN">
            <xs:selector xpath="./scl:DOI"/>
            <xs:field xpath="@name"/>
        </xs:unique>
        <xs:unique name="uniqueLogInLN">
            <xs:selector xpath="./scl:Log"/>
            <xs:field xpath="@name"/>
        </xs:unique>
    </xs:element>
    <xs:complexType name="tAnyLN" abstract="true">
        <xs:complexContent>
            <xs:extension base="tUnNaming">
                <xs:sequence>
                    <xs:element name="DataSet" type="tDataSet" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="ReportControl" type="tReportControl" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="LogControl" type="tLogControl" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="DOI" type="tDOI" minOccurs="0" maxOccurs="unbounded">
                        <xs:unique name="uniqueSDI_DAIinDOI">
                            <xs:selector xpath="./scl:DAI|./scl:SDI"/>
                            <xs:field xpath="@name"/>
                            <xs:field xpath="@ix"/>
                        </xs:unique>
                    </xs:element>
```

```
            <xs:element name="Inputs" type="tInputs" minOccurs="0">
              <!--<xs:unique name="uniqueExtRefInInputs">
                <xs:selector xpath="./scl:ExtRef"/>
                <xs:field xpath="@iedName"/>
                <xs:field xpath="@ldInst"/>
                <xs:field xpath="@prefix"/>
                <xs:field xpath="@lnClass"/>
                <xs:field xpath="@lnInst"/>
                <xs:field xpath="@doName"/>
                <xs:field xpath="@daName"/>
                <xs:field xpath="@intAddr"/>
              </xs:unique>-->
            </xs:element>
            <xs:element name="Log" type="scl:tLog" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="lnType" type="tName" use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tLN">
      <xs:complexContent>
        <xs:extension base="tAnyLN">
          <xs:attribute name="prefix" type="tPrefix" use="optional" default=""/>
          <xs:attribute name="lnClass" type="tLNClassEnum" use="required"/>
          <xs:attribute name="inst" type="tLNInst" use="required"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tLN0">
      <xs:complexContent>
        <xs:extension base="tAnyLN">
          <xs:sequence>
            <xs:element name="GSEControl" type="tGSEControl" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="SampledValueControl" type="tSampledValueControl" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="SettingControl" type="tSettingControl" minOccurs="0"/>
          </xs:sequence>
```

```
            <xs:attribute name="lnClass" type="tLNClassEnum" use="required" fixed="LLN0"/>
            <xs:attribute name="inst" type="xs:normalizedString" use="required" fixed=""/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tDataSet">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:choice maxOccurs="unbounded">
                <xs:element name="FCDA" type="tFCDA"/>
            </xs:choice>
            <xs:attribute name="name" type="tDataSetName" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tFCDA">
    <xs:attribute name="ldInst" type="tLDInst" use="optional"/>
    <xs:attribute name="prefix" type="tPrefix" use="optional" default=""/>
    <xs:attribute name="lnClass" type="tLNClassEnum" use="optional"/>
    <xs:attribute name="lnInst" type="tLNInst" use="optional"/>
    <xs:attribute name="doName" type="tFullDOName" use="optional"/>
    <xs:attribute name="daName" type="tFullAttributeName" use="optional"/>
    <xs:attribute name="fc" type="tFCEnum" use="required"/>
    <xs:attribute name="ix" type="xs:unsignedInt" use="optional"/>
</xs:complexType>
<xs:complexType name="tControl" abstract="true">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:attribute name="name" type="tCBName" use="required"/>
            <xs:attribute name="datSet" type="tDataSetName" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tControlWithTriggerOpt" abstract="true">
    <xs:complexContent>
        <xs:extension base="tControl">
```

```
            <xs:sequence>
                <xs:element name="TrgOps" type="tTrgOps" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="intgPd" type="xs:unsignedInt" use="optional" default="0"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tTrgOps">
    <xs:attribute name="dchg" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="qchg" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="dupd" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="period" type="xs:boolean" use="optional" default="false"/>
    <xs:attribute name="gi" type="xs:boolean" use="optional" default="true"/>
</xs:complexType>
<xs:complexType name="tReportControl">
    <xs:complexContent>
        <xs:extension base="tControlWithTriggerOpt">
            <xs:sequence>
                <xs:element name="OptFields">
                    <xs:complexType>
                        <xs:attributeGroup ref="agOptFields"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="RptEnabled" type="tRptEnabled" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="rptID" type="tMessageID" use="optional"/>
            <xs:attribute name="confRev" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="buffered" type="xs:boolean" use="optional" default="false"/>
            <xs:attribute name="bufTime" type="xs:unsignedInt" use="optional" default="0"/>
            <xs:attribute name="indexed" type="xs:boolean" use="optional" default="true"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tRptEnabled">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
```

```
            <xs:sequence>
                <xs:element name="ClientLN" type="tClientLN" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="max" use="optional" default="1">
                <xs:simpleType>
                    <xs:restriction base="xs:unsignedInt">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tClientLN">
    <xs:attributeGroup ref="agLNRef"/>
    <xs:attribute name="apRef" type="tAccessPointName" use="optional"/>
</xs:complexType>
<xs:complexType name="tLogControl">
    <xs:complexContent>
        <xs:extension base="tControlWithTriggerOpt">
            <xs:attribute name="ldInst" type="tLDInst" use="optional"/>
            <xs:attribute name="prefix" type="tPrefix" use="optional" default=""/>
            <xs:attribute name="lnClass" type="tLNClassEnum" use="optional" default="LLN0"/>
            <xs:attribute name="lnInst" type="tLNInst" use="optional"/>
            <xs:attribute name="logName" type="tLogName" use="required"/>
            <xs:attribute name="logEna" type="xs:boolean" use="optional" default="true"/>
            <xs:attribute name="reasonCode" type="xs:boolean" use="optional" default="true"/>
            <xs:attribute name="bufTime" type="xs:unsignedInt" use="optional" default="0"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tInputs">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="ExtRef" type="tExtRef" maxOccurs="unbounded"/>
```

```
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tExtRef">
    <xs:attributeGroup ref="scl:agDesc"/>
    <xs:attribute name="iedName" type="tIEDNameOrRelative" use="optional"/>
    <xs:attribute name="ldInst" type="tLDInst" use="optional"/>
    <xs:attribute name="prefix" type="tPrefix" use="optional"/>
    <xs:attribute name="lnClass" type="tLNClassEnum" use="optional"/>
    <xs:attribute name="lnInst" type="tLNInst" use="optional"/>
    <xs:attribute name="doName" type="tFullDOName" use="optional"/>
    <xs:attribute name="daName" type="tFullAttributeName" use="optional"/>
    <xs:attribute name="intAddr" type="xs:normalizedString" use="optional"/>
    <xs:attribute name="serviceType" type="tServiceType" use="optional"/>
    <xs:attribute name="srcLDInst" type="tLDInst" use="optional"/>
    <xs:attribute name="srcPrefix" type="tPrefix" use="optional"/>
    <xs:attribute name="srcLNClass" type="tLNClassEnum" use="optional"/>
    <xs:attribute name="srcLNInst" type="tLNInst" use="optional"/>
    <xs:attribute name="srcCBName" type="tCBName" use="optional"/>
    <xs:attribute name="pServT" type="tServiceType" use="optional"/>
    <xs:attribute name="pLN" type="tLNClassEnum" use="optional"/>
    <xs:attribute name="pDO" type="tFullDOName" use="optional"/>
    <xs:attribute name="pDA" type="tFullAttributeName" use="optional"/>
</xs:complexType>
<xs:complexType name="tLog">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:attribute name="name" type="tLogName" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tControlWithIEDName">
    <xs:complexContent>
        <xs:extension base="tControl">
            <xs:sequence>
```

```
                    <xs:element name="IEDName" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:simpleContent>
                                <xs:extension base="tIEDName">
                                    <xs:attribute name="apRef" type="tAccessPointName" use="optional"/>
                                    <xs:attribute name="ldInst" type="tLDInst" use="optional"/>
                                    <xs:attribute name="prefix" type="tPrefix" use="optional"/>
                                    <xs:attribute name="lnClass" type="tLNClassEnum" use="optional"/>
                                    <xs:attribute name="lnInst" type="tLNInst" use="optional"/>
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="confRev" type="xs:unsignedInt" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tProtocol">
        <xs:simpleContent>
            <xs:extension base="xs:normalizedString">
                <xs:attribute name="mustUnderstand" type="xs:boolean" use="required" fixed="true"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="tGSEControl">
        <xs:complexContent>
            <xs:extension base="tControlWithIEDName">
                <xs:sequence>
                    <xs:element name="Protocol" type="tProtocol" fixed="R-GOOSE" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="type" type="tGSEControlTypeEnum" use="optional" default="GOOSE"/>
                <xs:attribute name="appID" type="tMessageID" use="required"/>
                <xs:attribute name="fixedOffs" type="xs:boolean" use="optional" default="false"/>
                <xs:attribute name="securityEnable" type="scl:tPredefinedTypeOfSecurityEnum" use="optional"
default="None"/>
            </xs:extension>
```

```
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tSampledValueControl">
        <xs:complexContent>
            <xs:extension base="tControlWithIEDName">
                <xs:sequence>
                    <xs:element name="SmvOpts">
                        <xs:complexType>
                            <xs:attributeGroup ref="agSmvOpts"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="Protocol" type="tProtocol" fixed="R-SV" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="smvID" type="tMessageID" use="required"/>
                <xs:attribute name="multicast" type="xs:boolean" default="true"/>
                <xs:attribute name="smpRate" type="xs:unsignedInt" use="required"/>
                <xs:attribute name="nofASDU" type="xs:unsignedInt" use="required"/>
                <xs:attribute name="smpMod" type="tSmpMod" use="optional" default="SmpPerPeriod"/>
                <xs:attribute name="securityEnable" type="tPredefinedTypeOfSecurityEnum" use="optional"
default="None"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tSettingControl">
        <xs:complexContent>
            <xs:extension base="tUnNaming">
                <xs:attribute name="numOfSGs" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:unsignedInt">
                            <xs:minInclusive value="1"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="actSG" use="optional" default="1">
                    <xs:simpleType>
                        <xs:restriction base="xs:unsignedInt">
                            <xs:minInclusive value="1"/>
```

```
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="resvTms" type="xs:unsignedShort" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tDOI">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="SDI" type="tSDI">
                    <xs:unique name="uniqueSDI_DAIinSDI">
                        <xs:selector xpath="./scl:DAI|./scl:SDI"/>
                        <xs:field xpath="@name"/>
                        <xs:field xpath="@ix"/>
                    </xs:unique>
                </xs:element>
                <xs:element name="DAI" type="tDAI"/>
            </xs:choice>
            <xs:attribute name="name" type="tDataName" use="required"/>
            <xs:attribute name="ix" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="accessControl" type="xs:normalizedString" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tSDI">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element name="SDI" type="tSDI"/>
                <xs:element name="DAI" type="tDAI"/>
            </xs:choice>
            <xs:attribute name="name" type="tAttributeNameEnum" use="required"/>
            <xs:attribute name="ix" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="sAddr" type="xs:normalizedString" use="optional"/>
```

```
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tDAI">
    <xs:complexContent>
        <xs:extension base="tUnNaming">
            <xs:sequence>
                <xs:element name="Val" type="tVal" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="name" type="tAttributeNameEnum" use="required"/>
            <xs:attribute name="sAddr" type="xs:normalizedString" use="optional"/>
            <xs:attribute name="valKind" type="tValKindEnum" use="optional"/>
            <xs:attribute name="ix" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="valImport" type="xs:boolean" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tServiceYesNo"/>
<xs:complexType name="tServiceWithOptionalMax">
    <xs:attribute name="max" type="xs:unsignedInt" use="optional"/>
</xs:complexType>
<xs:complexType name="tServiceWithMax">
    <xs:attribute name="max" type="xs:unsignedInt" use="required"/>
</xs:complexType>
<xs:complexType name="tServiceWithMaxNonZero">
    <xs:attribute name="max" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:unsignedInt">
                <xs:minExclusive value="0"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:complexType>
<xs:complexType name="tServiceConfReportControl">
    <xs:complexContent>
        <xs:extension base="tServiceWithMax">
```

```
            <xs:attribute name="bufMode" use="optional">
                <xs:simpleType>
                    <xs:restriction base="xs:Name">
                        <xs:enumeration value="unbuffered"/>
                        <xs:enumeration value="buffered"/>
                        <xs:enumeration value="both"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="bufConf" type="xs:boolean" use="optional" default="false"/>
            <xs:attribute name="maxBuf" type="xs:unsignedInt" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tServiceWithMaxAndMaxAttributes">
    <xs:complexContent>
        <xs:extension base="tServiceWithMax">
            <xs:attribute name="maxAttributes" use="optional">
                <xs:simpleType>
                    <xs:restriction base="xs:unsignedInt">
                        <xs:minExclusive value="0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tServiceWithMaxAndModify">
    <xs:complexContent>
        <xs:extension base="tServiceWithMax">
            <xs:attribute name="modify" type="xs:boolean" use="optional" default="true"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="tServiceForConfDataSet">
    <xs:complexContent>
```

```
            <xs:extension base="tServiceWithMaxAndMaxAttributes">
                <xs:attribute name="modify" type="xs:boolean" use="optional" default="true"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="tClientServices">
        <xs:sequence>
            <xs:element name="TimeSyncProt" type="scl:tTimeSyncProt" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="goose" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="gsse" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="bufReport" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="unbufReport" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="readLog" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="sv" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="supportsLdName" type="xs:boolean" use="optional" default="false"/>
        <xs:attribute name="maxAttributes" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:unsignedInt"/>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="maxReports" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:unsignedInt"/>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="maxGOOSE" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:unsignedInt"/>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="maxSMV" use="optional">
            <xs:simpleType>
                <xs:restriction base="xs:unsignedInt"/>
            </xs:simpleType>
        </xs:attribute>
```